Consultative Committee for Space Data Systems

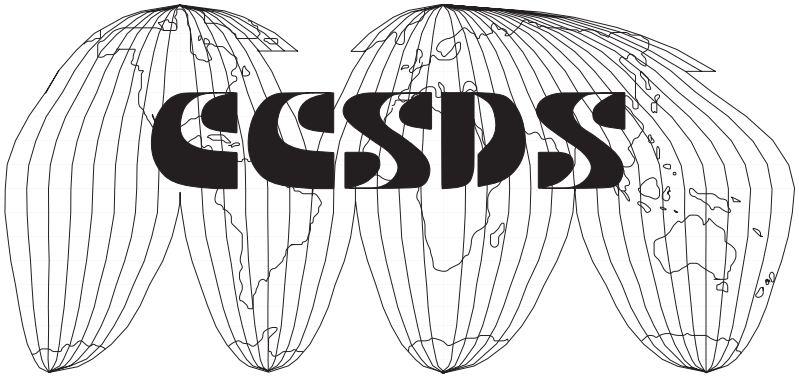
DRAFT RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

SPACE LINK EXTENSION—FORWARD SPACE PACKET SERVICE SPECIFICATION

CCSDS 912.3-R-1

RED BOOK

November 1997



AUTHORITY

Issue:	Red Book, Issue 1
Date:	November 1997
Location:	Oxfordshire, England

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS Recommendations is detailed in reference [F1], and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code MG)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

- Whenever an Agency establishes a CCSDS-related **standard**, the Agency will provide other CCSDS member Agencies with the following information:

  -- The **standard** itself.

  -- The anticipated date of initial operational capability.

  -- The anticipated duration of operational service.

- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommendation** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or, (3) be retired or canceled.

In those instances when a new version of a **Recommendation** is issued, existing CCSDS-related Agency standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each Agency to determine when such standards or implementations are to be modified. Each Agency is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommendation.

FOREWORD

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING FOREWORD:)

This document is a technical **Recommendation** for use in developing ground systems for space missions and has been prepared by the **Consultative Committee for Space Data Systems** (CCSDS). The Space Link Extension Forward Space Packet Service described herein is intended for missions that are cross-supported between Agencies of the CCSDS.

This **Recommendation** specifies a data service that extends certain of the space-to-ground communications services previously defined by CCSDS (references [2], [3], and [4]) within the framework established by the CCSDS Space Link Extension Reference Model (reference [1]). It allows implementing organizations within each Agency to proceed with the development of compatible, derived Standards for the ground systems that are within their cognizance. Derived Agency Standards may implement only a subset of the optional features allowed by the **Recommendation** and may incorporate features not addressed by the **Recommendation**.

Through the process of normal evolution, it is expected that expansion, deletion or modification to this document may occur. This **Recommendation** is therefore subject to CCSDS document management and change control procedures, as defined in reference [F1].

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.
- Russian Space Agency (RSA)/Russian Federation.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Federal Service of Scientific, Technical & Cultural Affairs (FSST&CA)/Belgium.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Centre (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Korea Aerospace Research Institute (KARI)/Korea
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 912.3-R-1	Space Link Extension—Forward Space Packet Service Specification	November 1997	Original Issue

CONTENTS

CONTENTS (continued)

Section Page

Figure

Table

1 INTRODUCTION

1.1 PURPOSE OF THIS RECOMMENDATION

This Recommendation defines the Forward Space Packet (FSP) service conformant to the transfer services specified in reference [1], *Cross Support Reference Model—Part 1: Space Link Extension Services*. This service specifies a provisioning of the FSP service as defined in 3.4.1.3 b of the Reference Model.

1.2 SCOPE

This Recommendation defines, in an abstract manner, the Forward Space Packet service within the application layer in terms of:

a) the actions and events of the operations necessary to provide the transfer service;

b) the parameter data associated with each service operation's action and events;

c) the behaviors that result from the invocation of each operation;

d) the relationship between, and the valid sequence of, the operations and resulting behaviors.

It does not specify:

a) individual implementations or products;

b) the implementation of entities or interfaces within real systems;

c) the methods or technologies required to radiate telecommand packets to the spacecraft and to acquire telemetry frames from the signals received from the spacecraft for extraction of the Operational Control Field;

d) the methods or technologies required to provide a suitable environment for communications;

e) the management activities required to schedule, configure, and control the FSP service.

1.3 APPLICABILITY

1.3.1 APPLICABILITY OF THIS RECOMMENDATION

This Recommendation provides a basis for the development of real Systems that implement the FSP service. Implementation of the FSP service in a real system additionally requires the availability of a communications service to convey invocations and returns of FSP service operations between FSP service users and providers. This Recommendation requires that

such a communications service ensure that invocations and returns of operations are transferred:

a) in sequence;

b) completely and with integrity;

c) without duplication;

d) with flow control;

e) with notification to the application layer in the event that communications between the FSP service user and the FSP service provider are disrupted.

It is the specific intent of this Recommendation to define the FSP service in a manner that is independent of any particular communications services, protocols, or technologies.

1.3.2 LIMITS OF APPLICABILITY

This Recommendation specifies the FSP service that may be provided by an SLE System for inter-agency cross support. It is neither a specification of, nor a design for, real SLE systems that may be implemented for the control and monitoring of existing or future missions.

1.4 RATIONALE

The goal of this Recommendation is to increase the level of interoperability between the tracking stations or ground data handling systems of various agencies and the users of commanding services.

1.5 DOCUMENT STRUCTURE

This Recommendation is organized as follows:

a) Section 1 provides purpose, scope, applicability and rationale of this Recommendation, and lists the definitions, conventions, and references used throughout the Recommendation.

b) Section 2 provides an overview of the FSP service including a functional description, the service management context, and protocol considerations.

c) Section 3 specifies the operations of the FSP service.

d) Section 4 specifies the dynamic behavior of the FSP service in terms of FSP service provider state transitions.

e) Annex A provides a formal specification of FSP service data types using the Abstract Syntax Notation One (ASN.1).

f) Annex B provides a specification of the multiplexing between concurrent FSP service instances sharing the same telecommand Virtual Channel as well as the multiplexing between telecommand Virtual Channels sharing the same physical space link channel.

g) Annex C lists all terms used in this Recommendations and identifies where they are defined.

h) Annex D lists all acronyms used within this document.

i) Annex E provides a conformance matrix that defines what capabilities must be provided for an implementation to be considered compliant with this Recommendation.

j) Annex F contains a list of informative references.

1.6 DEFINITIONS, NOMENCLATURE, AND CONVENTIONS

1.6.1 DEFINITIONS

1.6.1.1 Definitions from the OSI Basic Reference Model

This Recommendation makes use of a number of terms defined in reference [7]. The use of those terms in this Recommendation shall be understood in a generic sense; i.e., in the sense that those terms are generally applicable to technologies that provide for the exchange of information between real systems. Those terms are:

a) abstract syntax;

b) application entity;

c) application layer;

d) application process;

e) application process invocation;

f) concatenation;

g) concrete syntax;

h) flow control;

i) Open Systems Interconnection (OSI);

j) real system;

k) service access point (SAP);

l) transfer syntax.

1.6.1.2 Definitions from Abstract Syntax Notation One

This Recommendation makes use of the following terms defined in reference [5]:

a) Abstract Syntax Notation One (ASN.1);

b) object identifier;

c) (data) type;

d) (data) value.

NOTE – In annex A of this Recommendation, ASN.1 is used for specifying the abstract syntax of FSP service operations. The use of ASN.1 as a descriptive language is intended to support the specification of the abstract FSP service; it is not intended to constrain implementations. In particular, there is no requirement for implementations to employ ASN.1 encoding rules. ASN.1 is simply a convenient tool for formally describing the abstract syntax of the FSP service operations.

1.6.1.3 Definitions from Telecommand Channel Coding

This Recommendation makes use of the following term defined in reference [2]: Command Link Transmission Unit.

1.6.1.4 Definitions from Telecommand Routing Service

This Recommendation makes use of the following terms defined in reference [3]:

a) Command Link Control Word;

b) Command Operation Procedure;

c) Frame Operation Procedure;

d) Multiplexer Access Point;

e) Receiver_Frame_Sequence_Number V(R);

f) Telecommand Segment;

g) Telecommand Transfer Frame (TC Transfer Frame or TC Frame);

h) Virtual Channel.

1.6.1.5 Definitions from Telecommand Command Operation Procedure

This Recommendation makes use of the following terms defined in reference [4]:

a) AD, BD, BC;

b) Suspend_State;

c) Timeout_Type;

d) T1_Initial;

e) Receiver_Frame_Sequence_Number V(R).

1.6.1.6 Definitions from SLE Reference Model

This Recommendation makes use of the following terms defined in reference [1]:

a) abstract binding;

b) abstract object;

c) abstract port;

d) abstract service;

e) Forward Space Packet channel (FSP channel);

f) Forward Space Packet service (FSP service);

g) initiator;

h) invoker;

i) Mission Data Operation System (MDOS);

j) Mission User Entity (MUE);

k) offline delivery mode;

l) online delivery mode;

m) operation;

n) performer;

o) physical channel;

p) responder;

q) return data;

r) service user (user);

s) service provider (provider);

t) SLE Complex;

u) SLE Complex Management;

v) SLE data channel;

w) SLE functional group (SLE-FG);

x) SLE protocol data unit (SLE-PDU);

y) SLE service data unit (SLE-SDU);

z) SLE service package;

aa) SLE System;

bb) SLE transfer service instance;

cc) SLE transfer service production;

dd) SLE transfer service provision;

ee) SLE transfer service instance provision period;

ff) SLE Utilization Management;

gg) space link;

hh) space link data channel;

ii) space link data unit (SL-DU);

jj) space link session.

1.6.1.7 Additional Definitions

NOTE – For the purpose of this Recommendation, the following definitions also apply.

1.6.1.7.1 Application Context

An application context is technology-specific information that describes both the service being offered by the provider and the required communications environment.

NOTE – For example, some OSI-based applications use an application context name and a context description that are passed to an association control service element. The TCP/IP-based Distributed Computing Environment (DCE) uses a 128-bit integer to identify interfaces. Successful negotiation of the application context must occur between user and provider before an association is established.

1.6.1.7.2 Association

An association is a cooperative relationship between an SLE service-providing application process and an SLE service-using application process.

1.6.1.7.3 Communications Service

A communications service is a capability that enables an SLE service-providing application process invocation and an SLE service-using application process invocation to exchange information.

NOTE – If an SLE service user and an SLE service provider are implemented using different communications services, then interoperability between them is possible only by means of a suitable gateway. Adherence to this Recommendation ensures, at least in principle, that it is possible to construct such a gateway.

1.6.1.7.4 Confirmed Operation

A confirmed operation is an operation that requires the performer to return a report of its outcome to the invoker.

1.6.1.7.5 Invocation

The invocation of an operation is the making of a request by an object (the invoker) to another object (the performer) to carry out the operation.

1.6.1.7.6 Parameter

A parameter of an operation is data that may accompany the operation's invocation or return.

1.6.1.7.7 Performance

The performance of an operation is the carrying out of the operation by an object (the performer).

1.6.1.7.8 Port Identifier

A port identifier identifies a logical channel in a communications service. A port identifier contains all the addressing information necessary to make a connection possible.

NOTE – Port identifiers are technology-specific. For example, a port identifier might be the combination of an Internet Protocol (IP) network address and a Transmission Control Protocol (TCP) port number or the combination of an OSI network address and an associated set of service access points (SAPs).

1.6.1.7.9 Return

The return of an operation is a report, from the performer to the invoker, of the outcome of the performance of the operation.

1.6.1.7.10 Unconfirmed Operation

An unconfirmed operation is an operation that does not require a report of its outcome to be returned to the invoker by the performer.

1.6.2 NOMENCLATURE

The following conventions apply throughout this Recommendation:

a) the words ‘shall’ and ‘must’ imply a binding and verifiable specification;

b) the word ‘should’ implies an optional, but desirable, specification;

c) the word ‘may’ implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

1.6.3 CONVENTIONS

1.6.3.1 Specification of Operations

1.6.3.1.1 General

Section 3 of this Recommendation specifies the operations that comprise the FSP service. The specification of each operation is divided into subsections as follows:

1.6.3.1.2 Purpose Subsection

The Purpose subsection provides a brief description of the purpose of the operation. Additionally, it indicates whether the operation may be invoked by the user, provider, or both; whether the operation is confirmed or unconfirmed; and whether there are any constraints on when the operation may be invoked.

1.6.3.1.3 Invocation, Return, and Parameters Subsection

The Invocation, Return, and Parameters subsection describes the parameters associated with each operation, including their semantics. A table accompanying the description of each operation lists all parameters associated with the operation and, for both the invocation and return, whether the parameter is always present, always absent, or conditionally present.

For parameters that are conditionally present, the parameter description specifies the conditions for the presence or absence of the parameter. The condition is generally based on the value of another parameter in the same invocation or return; for example, the parameter `diagnostic` is present in a return if and only if the value of the parameter `result` is 'negative result'. For a conditional parameter in a return, the condition may be based on the value of a parameter in the corresponding invocation.

In the table, the following convention is used to indicate whether a parameter is always present, always absent, or conditionally present:

M	Always present
C	Conditionally present
Blank	Always absent

NOTE – Even though a parameter may be characterized as always present, its description may specify that its value is permitted to be 'null' or 'unused' or the like.

1.6.3.1.4 Effects Subsection

The Effects subsection describes the effects an operation has on the invoker, the performer, the association between them, or any combination thereof. The details of how those effects occur or the mechanisms used are outside the scope of this Recommendation.

1.6.3.2 Typographic Conventions

1.6.3.2.1 Operation Names

Names of FSP service operations always appear in uppercase and begin with the characters 'FSP-' (e.g., FSP-TRANSFER-DATA).

1.6.3.2.2 Parameter Names

Names of parameters of FSP service operations generally appear in lowercase and are always typeset in a fixed-width font (e.g., `initiator-port-identifier`). When the value of a parameter is referred to rather than its name, the value is typeset in the normal text font.

1.6.3.2.3 Value and Type Names

Each parameter of each FSP service operation is specified as being of a certain type. The type of a parameter constrains the values that may be assigned to it. Values are always shown in quotation marks (e.g., 'no such service instance').

NOTE – The name of a value does not imply anything about type. For example, the value 'no such service instance' has the appearance of a character string but might be assigned to a parameter whose type is 'Integer'. Parameter types are specified in annex A.

1.6.3.2.4 State Names

This Recommendation specifies the states of FSP service providers. States may be referred to by number (e.g., state 2) or by name. State names are always shown in quotation marks (e.g., 'active').

1.6.3.2.5 Data Type Definitions

Data type definitions for the FSP service are presented in annex A in the form of a set of ASN.1 modules. Regardless of the conventions used elsewhere in this Recommendation, the text of the ASN.1 modules is typeset entirely in a fixed-width font.

1.6.3.3 Other Conventions

This Recommendation uses the conventions specified in reference [1].

1.7 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommendations.

[1] *Cross Support Reference Model—Part 1: Space Link Extension Services*. CCSDS 910.4-B-1. Blue Book. Issue 1. Pasadena, California, USA: CCSDS, May 1996.

[2] *Telecommand Part 1 — Channel Service*. Recommendation for Space Data System Standards, CCSDS 201.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 1995.

[3] *Telecommand Part 2 — Data Routing Service*. Recommendation for Space Data Systems Standards, CCSDS 202.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 1992.

[4] *Telecommand Part 2.1 — Command Operation Procedures*. Recommendation for Space Data System Standards, CCSDS 202.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, October 1991.

[5] *Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*. International Standard, ISO 8824. 2nd ed. Geneva: ISO, 1990.

[6] *Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[7] *Information technology—Text Communication—Message-Oriented Text Interchange Systems (MOTIS) —Part 3: Abstract Service Definition Conventions*. International Standard, ISO/IEC 10021-3:1990. Geneva: ISO, 1990.

2 DESCRIPTION OF THE FORWARD SPACE PACKET SERVICE

2.1 OVERVIEW

The FSP service enables the user of the service to send telecommand packets to a spacecraft. The FSP service user submits packets encapsulated in SLE Service Data Units (SLE-SDUs), by means of the FSP-TRANSFER-DATA operation.

The FSP service provider checks the packet header to determine if the packet sent by the user complies with the applicable constraints. Besides that, the service provider does not modify, interpret, or interrogate the contents of the packet but transmits it bit for bit as received from the service user.

The FSP service encapsulates CCSDS TC packets into blocks of CCSDS TC Frames (i.e., one or more frames) and passes these to an underlying Forward TC Frame (FTCF) service.

The operations defined in section 3 of this Recommendation enable an FSP service user to interact with an FSP service provider to:

a) establish an association between the user and the provider;

b) send annotated telecommand packets to the provider;

c) obtain notifications and reports regarding the status, configuration, and performance of the service;

d) temporarily suspend and later re-start the sending of telecommand packets;

e) set parameters that affect certain aspects of the behavior of the service;

f) release an association.

The FSP service is provided in the online delivery mode, as defined in reference [1]. The offline delivery mode is the subject of further study.

The provision of FSP service for access to one telecommand virtual channel (VC) by one service user constitutes one instance of service. The provision of FSP service for access to one virtual channel by multiple service users, and the provision of FSP service for access to multiple virtual channels by one or more service users are permitted, but are specified to constitute multiple service instances.

2.2 SPACE LINK EXTENSION REFERENCE MODEL

2.2.1 INTRODUCTION

The FSP service is specified within the framework defined by the SLE reference model (reference [1]). The SLE reference model is based, in part, on the Abstract Service Definition Conventions (ASDC) defined in reference [7]. The following paragraphs summarize selected concepts from the SLE reference model.

2.2.2 ABSTRACT OBJECT

An abstract object is a functional entity that interacts with other abstract objects. Objects are of different types, which determine their function and behavior. Objects are characterized by their interfaces, which are called abstract ports, and the operations which are made available through those interfaces (reference [1], 1.6.1.1).

2.2.3 ABSTRACT SERVICE

An abstract service is the capability provided by a set of operations that an abstract object exposes at one or more of its abstract ports (reference [1], 1.6.1.2).

NOTE – The concept of an abstract service is to be distinguished from the concept of an (N)-service as defined in the OSI Basic Reference Model (reference [6]). The definition of (N)-service is in terms of the capability provided by one layer in the OSI architecture to the layer above it. The definition of abstract service is in terms of the capability provided by one abstract object to another abstract object. In a cross support scenario, where one Agency is providing an SLE service to another Agency, the object that provides the service typically is associated with one Agency, and the object that uses the service typically is associated with the other Agency.

2.2.4 SERVICE USER/PROVIDER

An object that offers a service to another by means of one or more of its ports is called a service provider (provider). The other object is called a service user (user). An object may be a provider of some services and a user of others (reference [1], 1.6.1.6).

The terms user and provider are used to distinguish the roles of two interacting objects. In this Recommendation, when two objects are involved in provision of a service, the object closer to the space link is considered to be the provider of the service, and the object further from the space link is considered to be the user.

2.2.5 ABSTRACT BINDING

When two abstract ports of the same type have an association established between them such that an abstract service can be provided, the two ports are said to be bound. The act of establishing such an association is called abstract binding (reference [1], 1.6.1.7).

The terms initiator and responder are used to describe the interaction between two objects with respect to abstract binding. One object (the initiator) invokes a bind operation which is accepted (or rejected) by another object (the responder).

2.2.6 OPERATION

An operation is a procedure or task that one object (the invoker) can request of another (the performer) through a port pair bound within the terms of an agreement (reference [1], 1.6.1.8).

The terms invoker and performer are used to describe the interaction between two objects as the operations that constitute the service occur. One object invokes an operation that is performed by the other. For most services, each object invokes some operations and performs others.

2.3 SERVICE MANAGEMENT

SLE service management determines the number, type and schedule of FSP service instances to be provided, the resources required to enable those service instances, and the initial configuration of all service instances and their supporting resources. SLE service management is the subject of separate CCSDS Recommendations, and the CCSDS Secretariat should be contacted for information regarding the status of those Recommendations.

The SLE reference model (reference [1]) distinguishes between service provision and service production. Certain configuration parameters are associated with provision of FSP service while others are associated with production. Changes to FSP provision configuration parameters affect only a single service instance. The values of such parameters are initialized by service management when the service instance is created but may be modified subsequently by the user through FSP service operations specified in this Recommendation. Changes to FSP production configuration parameters (e.g., maximum packet size, maximum segment size, multiplexing scheme) potentially affect multiple service instances or potentially impact SLE Complex resources; consequently, those parameters are modified only through service management. However, although the invocation of FOP directives will in general affect multiple service instances, they can be invoked by one service instance per VC. Some affect also the space element and are therefore regarded to be integral parts of a telecommand service and not parts of the management of such service.

FSP service is user-initiated (i.e., the user invokes the bind operation).

2.4 ARCHITECTURE MODEL—FUNCTIONAL VIEW

2.4.1 FORWARD TC-VC DATA INSERTION FUNCTIONAL GROUP

The Forward TC-VC Data Insertion Functional Group (shown in figure 2-1) is the SLE functional group (SLE-FG) that produces the FSP service and the Forward TC-VCA service. The latter service is not addressed by this Recommendation but is to be defined in a companion Recommendation.

As described in reference [1], the Forward TC VC Data Insertion Functional Group consumes one or more Forward Space Packet data channels consisting of a stream of Forward Space Packet SLE-SDUs and supplies the TC Frame SLE data channel.

NOTE – As defined in reference [1], subsection 5.6.2.9.2, the production of the Forward Space Packet service involves three SLE Functional Groups: the Forward TC Space Link Processing SLE-FG, the Forward CLTU Generation SLE-FG, and the Forward TC VC Data Insertion SLE-FG.

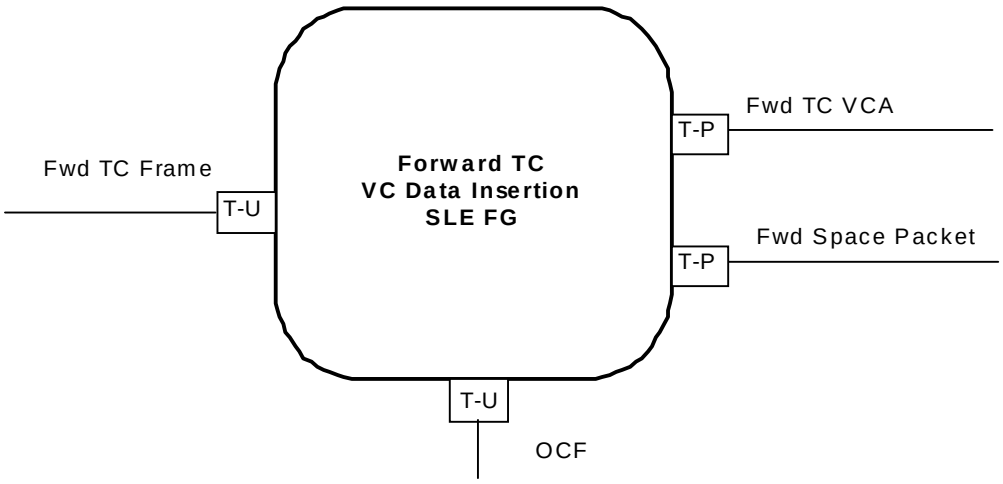


Figure 2-1: Forward TC VC Data Insertion SLE-FG

More specifically, the Forward TC VC Data Insertion Functional Group performs the following functions with respect to FSP service:

a) For each Forward Space Packet service instance, the Forward TC VC Data Insertion SLE-FG consumes one or more Forward Space Packet data channels, extracts the CCSDS packets from the Forward Space Packet SLE-SDUs in each data channel, blocks or segments the CCSDS packets into Telecommand Transfer Frames, and

supplies the TC Frame SLE data channel in the form of blocks of one or more TC transfer frames. If the Frame is to be sent using the Sequence-controlled service option of the Command Operation Procedure (COP), the TC VC Data Insertion SLE-FG sets the Frame Sequence Count and buffers a copy of the TC Transfer Frame.

b) The Forward TC VC Data Insertion SLE-FG consumes MC-OCF or VC-OCF SLE data channels and extracts the Command Link Control Words (CLCWs). Based on the values in the CLCW, the Forward TC VC Data Insertion SLE-FG retransmits or deletes buffered TC Transfer Frames.

c) The Forward TC VC Data Insertion SLE-FG supplies the TC frame SLE data channels to the Forward CLTU Generation FG through an online service instance.

The CCSDS packets consumed by the FSP service are encapsulated in SLE-SDUs. The FSP SLE-SDUs contain control and annotation data that pertains to that specific packet. This data consist of:

a) the time interval during which the packet is to be transmitted to the space element;

b) the minimum time between the CLTU(s) encapsulating this packet and a subsequent CLTU;

c) the transmission mode of this packet (sequence controlled or expedited);

d) if applicable, the Multiplexer Access Point (MAP) via which the packet will be routed;

e) the aggregation flag, indicating if the packet may or may not share a TC Frame with other packets;

f) the report selectors, indicating if notifications are to be generated when the packet has been radiated and/or transferred to the space element.

2.4.2 FSP SERVICE PRODUCTION AND PROVISION

SLE-SDUs are delivered to the service provider by means of the FSP service operations defined in section 3. The general relationship between SL-DUs, SLE-SDUs, and FSP service operations is illustrated in figure 2-2.

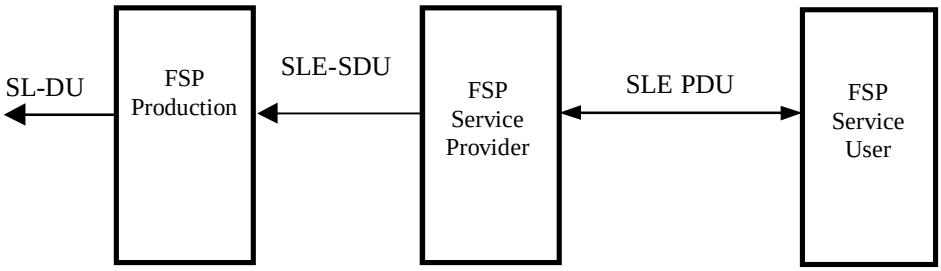


Figure 2-2: FSP Service Production and Provision

One instance of FSP service production (or, one instance of the Forward TC VC Data Insertion Functional Group) may be associated with multiple FSP service instances. FSP production is concerned with the extraction of CCSDS packets from concurrent streams of the SLE-SDUs and their multiplexing onto the space link according to the control and annotation of the packet in the SLE-SDU and the configuration set up by service management. This process is independent of any particular instance of service. In contrast, FSP service provision is concerned with receiving one stream of SLE-SDUs from an FSP service user. Service provision addresses such matters as when service is provided (e.g., service start and stop times), how service is provided (e.g., which events are notified to the user), and quality of service (e.g., whether directives may be invoked).

The SLE-SDUs consumed by FSP service production are sent by the service user by means of the FSP service operations defined in section 3, which also provide additional functionality to facilitate the provision of FSP service. In turn, the FSP service operations are realized as SLE protocol data units (SLE-PDUs) which are exchanged between the FSP service provider and the FSP service user by means of an underlying communications service. The general relationship between SL-DUs, SLE-SDUs, and SLE-PDUs is illustrated in figure 2-2.

Production of the FSP service by the provider occurs during the Space Link session. This will in general largely overlap with service provisioning.

2.5 ARCHITECTURE MODEL—CROSS SUPPORT VIEW

The management and control of the production and provisioning of the SLE transfer services is specified in reference [1]. Figure 2-3 shows an example operational scenario and the related binding of FSP transfer service ports (defined in this Recommendation) and the SLE management ports (outside the scope of this Recommendation). This scenario shows an SLE Complex with one Forward TC VC Data Insertion Functional Group instance providing two instances of FSP service to a Mission Data Operations System. The Forward TC VC Data Insertion Functional Group uses the FTCF service provided by a Forward CLTU Generation Functional Group which in turn uses the CLTU service provided by a Forward TC Space Link Processing Functional Group.

NOTE – Although not shown in this scenario, many other combinations are possible. For example, it is also possible to have several SLE-FG instances, each supplying a different physical channel and each providing one or more instances of service. The different SLE-FGs shown in the scenario may also belong to different SLE Complexes.

2.6 FUNCTIONAL DESCRIPTION

2.6.1 GENERAL

This subsection describes the FSP service with respect to scheduling, configuration, underlying services, provider states and protocol considerations.

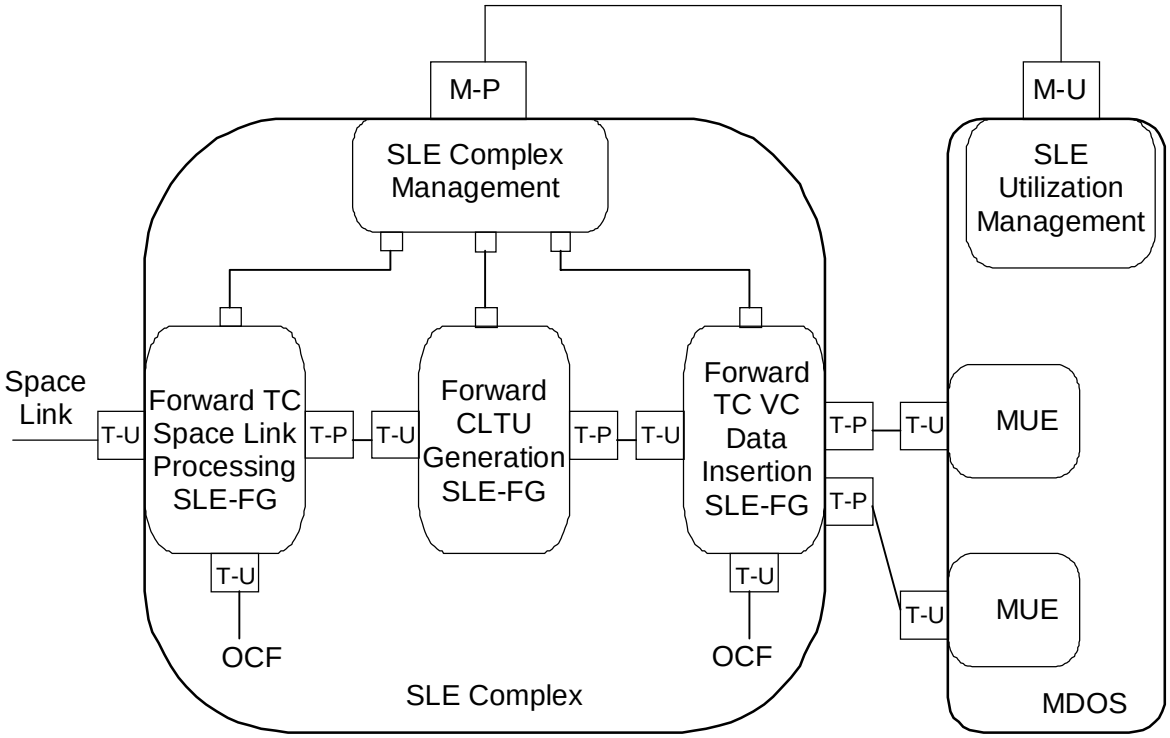


Figure 2-3: An Example of Binding of the FSP Service SLE Complex

2.6.2 SCHEDULING AND CONFIGURATION

SLE Utilization Management negotiates with SLE Complex Management to establish mutually agreed upon SLE service packages. Among other things, SLE service packages specify what service instances are to be provided, when those services are to be provided, and what resources are needed to enable those services.

Service packages also specify the initial values of mission-dependent parameters required for service production and provision. FSP service production parameters include such things as the telecommand VC to be used, the multiplexing schemes applicable at TC frame and MAP level, maximum packet and TC frame size. FSP service provision parameters include such things as the scheduled start and stop times of the FSP service instance.

Service production is guaranteed to occur only as needed to support service packages that have been scheduled and mutually agreed upon by SLE Complex Management and SLE Utilization Management. Service provision occurs only within the bounds of the agreed upon schedule of service instances and only during those periods when there is an association between the service provider and the service user.

2.6.3 UNDERLYING SERVICES

The FSP service relies on the following SLE services:

- Fwd TC Frame;
- Fwd CLTU.

A schedule for an FSP service instance must be compliant with the schedule of the underlying TC Frame service.

2.6.4 DELIVERY MODES

2.6.4.1 Forward Online Delivery

Forward online delivery service provisioning occurs at the same time as service production, i.e., during a Space Link Session. Packets supplied by the service user are buffered by the service provider until they are processed. The buffer used by the service provider exists only during service production. The packet processing time is determined by the order of packets in the buffer and any annotation data provided with the packets.

Two transmission modes are available to the service user:

a) sequence controlled;

b) expedited.

Sequence controlled transmission mode means that there is a high probability of complete delivery of the packet, that no packet is duplicated and that no packet is delivered out of sequence.

Expedited mode delivers packets in the sequence specified but does not guarantee complete delivery.

2.6.4.2 Forward Offline Delivery[1]

Service provisioning and service production do not overlap. Packets supplied by the service user during service provision are buffered by the provider in persistent storage until service production.

[1] The FSP service as specified in this Recommendation is restricted to online delivery mode. The offline delivery mode is the subject of further study.

2.6.5 PROTOCOL DESCRIPTION

2.6.5.1 States of the Service Provider

Once an FSP service instance is created, the FSP service provider is in one of three states, as follows:

a) State 1 (‘unbound’): In state 1, all resources required to enable the provision of the FSP service have been allocated, and all objects required to provide the service have been instantiated. However, no association yet exists between the user and the provider (i.e., the FSP transfer service provider port is not bound).

b) State 2 (‘ready’): In state 2, an association has been established between user and provider, and they may interact by means of the operations described in section 3 of this Recommendation. However, sending of telecommand packets (by means of the FSP-TRANSFER-DATA operation) from the user to the provider is not permitted. The user may enable the delivery of telecommand packets by means of the appropriate service operation (FSP-START), which, in turn, will cause the provider to transition to the active state and enable frame delivery.

c) State 3 (‘active’): State 3 is identical to the ready state except that now the user can send telecommand packets. The service continues in this state until the user invokes the FSP-STOP operation to suspend frame delivery and transition back to the ready state.

A simplified FSP service provider state transition diagram is shown in figure 2-4. A detailed state transition matrix is provided in section 4.

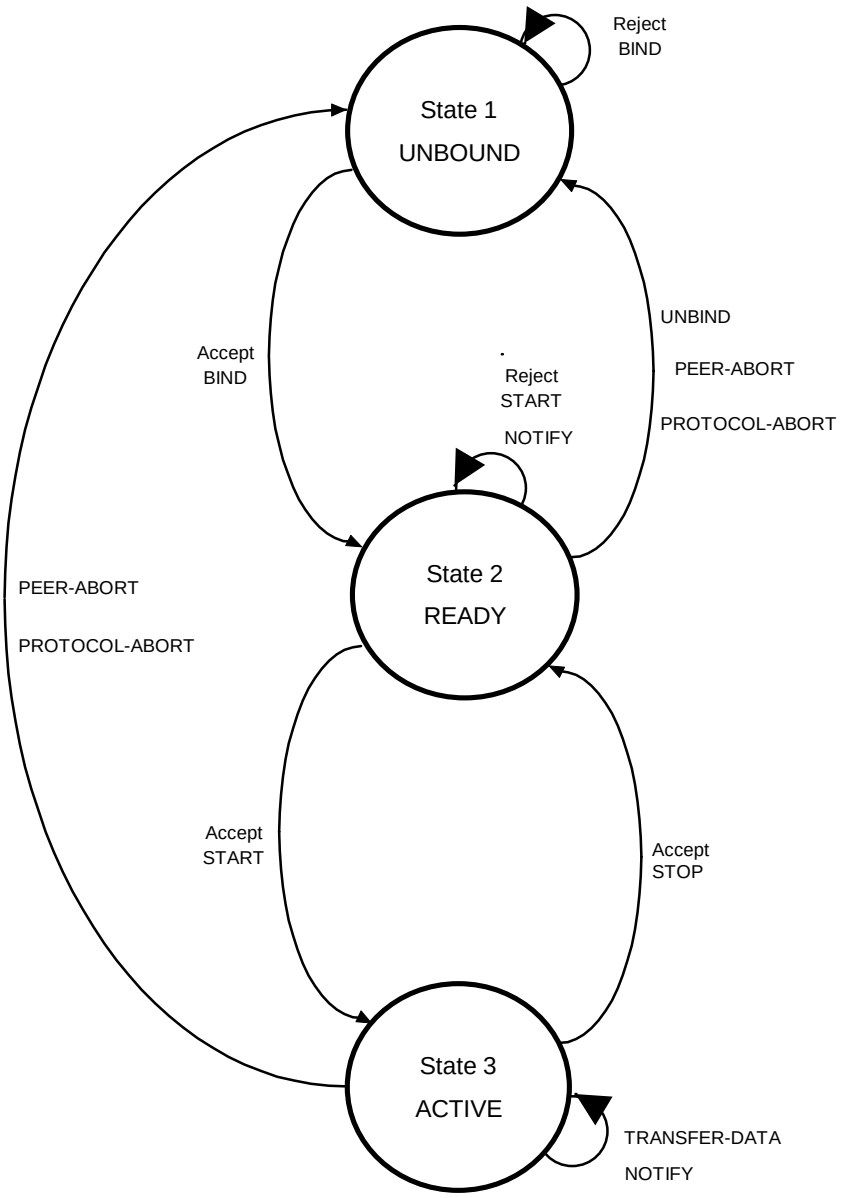


Figure 2-4: Simplified FSP Service Provider State Transition Diagram

2.6.5.2 Terminating an Association

An association is released normally when an FSP-UNBIND operation is invoked by the initiator of the association and performed by the responder. An association may be aborted by either the user or the provider by means of the FSP-PEER-ABORT operation. An association also may be aborted because of certain failures of the communications service. Such failures are signaled to the user and provider applications by the FSP-PROTOCOL-ABORT operation.

2.6.5.3 Effects of Association Termination

When an association is terminated or aborted, no further operations can be exchanged between the user and the provider. The provider discards all packets buffered for this service instance but completes the radiation of any packet that is already in the production process. The systems may reestablish an association via a new FSP-BIND operation. However, status information from the prior association is not preserved and is not available to the new association with the following exceptions: All accounting information, such as number of packets processed, is persistent during the service instance provision period. Similarly, parameters that serve to relate notifications on an activity to operations that triggered this activity will not be altered when the association is released or aborted.

2.6.5.4 Technology Aspects

This Recommendation defines the FSP service. Provision of the FSP service in a real system also requires a specification of how the FSP service defined here is mapped to a communications service such that invocations and returns of FSP service operations can be exchanged between the user and the provider. In order not to restrict the applicability of this Recommendation to a specific communications technology, as few assumptions as possible have been made about the characteristics of the underlying communications service (see 1.3.1).

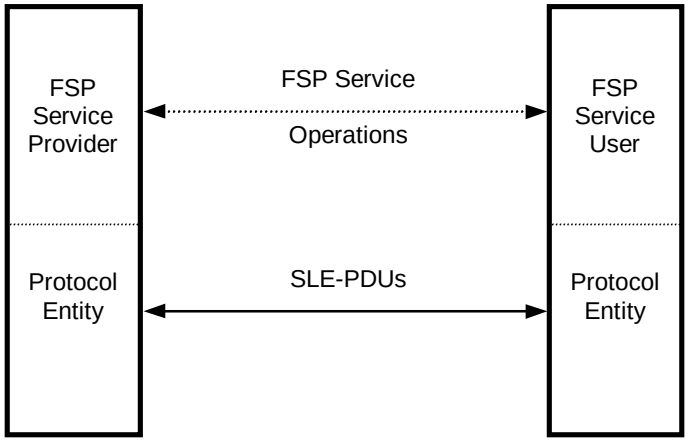


Figure 2-5: Mapping of FSP Service Operations to SLE-PDUs

The service interface between the user and the provider is specified in this Recommendation in terms of the operations that the service provides. However, implementation of the service in a real system requires a mapping of the service operations to protocol data units that can be exchanged by means of the communications service. Within the scope of this Recommendation, there is a conceptual distinction between FSP service operations and SLE-PDUs; in this version of the Recommendation no concatenation of multiple FSP operation

invocations into single SLE-PDU is foreseen. From the point of view of the provider or user application, the interaction is in terms of operations; but from the point of view of the application-entities that implement the protocol, what is exchanged are SLE-PDUs that may represent multiple FSP service operations. The mapping of service operations to SLE-PDUs is illustrated in figure 2-5.

The situation is complicated further when the underlying communications service is considered. In the case where the peer applications have been built on top of different communication technologies, it is still possible to achieve interoperability, but only by means of a gateway. Figure 2-6 shows a user and a provider that use different communications technologies and interoperate through a gateway. The FSP SLE-PDUs are translated on one side into forms appropriate for a particular communications technology. These forms are then translated by the gateway into forms appropriate for another communications technology. Finally, those forms are translated back into FSP SLE-PDUs.

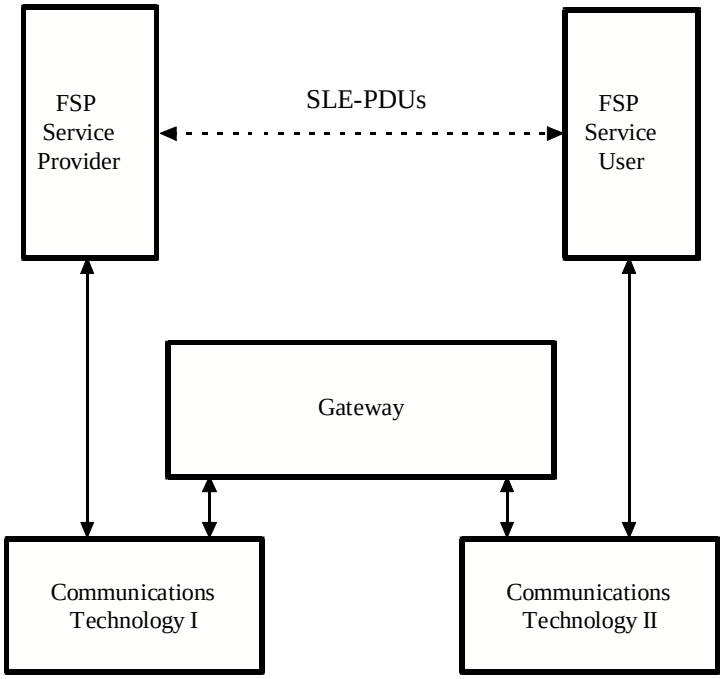


Figure 2-6: FSP Service Provision Via a Gateway

NOTE – In order to accomplish interoperability, even in the case of implementations using the same underlying communications technology, there is still the need to agree on how to map the SLE-PDUs into forms appropriate for that communications technology.

Because the operations of the FSP service are relatively simple, once an association is in place between the service user and the service provider, the technology specific elements involved in the exchange of SLE-PDUs are relatively minor. However, the way an association is established (i.e., the binding) tends to be specific to the communications technology in use. Nonetheless, the FSP-BIND and FSP-UNBIND operations as specified in this document are

intended to be 'technology neutral'. This neutrality necessitates some flexibility in the specification of the parameters of the bind operation. The semantics and even the type of some of the bind parameters may vary depending on the communications technology that is used; any such differences must be reconciled by the gateway. The following paragraphs discuss these technology-dependent parameters.

NOTE – Conceptually, it is useful to view the gateway as being part of either the SLE Complex or the MDOS. In terms of management, the parameters relevant for the binding are communicated in the form of the communications technology employed by the system that does not include the gateway. It is the responsibility of the gateway-hosting system to convert the parameters submitted by management to the 'local' form.

The `initiator-port-identifier` parameter identifies the port the responder will use to communicate with the initiator. The format and content are technology dependent. A communications gateway will intercept this parameter and replace it with the port-identifier the responder will use to communicate with the gateway. Similarly, the gateway will make use of `initiator-port-identifier` to route from the responder to the initiator.

The `responder-port-identifier` parameter identifies the port to which the initiator is requesting to bind. If a gateway is used, it is actually the port of the gateway. The gateway will intercept this parameter and convert it to the port-identifier of the actual responder.

NOTE – Some technologies and/or implementations do not disclose the actual 'physical' address of the application, but perform a mapping by means of a directory service. This is considered a local matter and must not be visible at the level of the service specified in this Recommendation.

The `application-context` parameter identifies, in a technology specific way, the peer application. It also identifies, either explicitly or implicitly, the abstract and transfer syntaxes, which, with some technologies, are then also subject to negotiation during the FSP-BIND operation. A gateway will need to intercept this parameter and convert it as necessary.

Binding is accomplished only when the association between the peer applications is established. A communications gateway implementation must ensure this. In particular, it must only return a report of the outcome of the FSP-BIND operation to the initiator when it has received such a return from the responder. It is not sufficient to establish an association between the initiator and the gateway.

A gateway must also ensure the consistency of the state of the application protocol on the user and the provider sides. In particular, in case the association with the FSP application on one side of the gateway is aborted (e.g., because of a communications service fault), the gateway must ensure that the association with the FSP application on the other side of the gateway is also aborted.

2.7 OPERATIONAL SCENARIO

The following paragraphs illustrate a typical sequence of operations between the user and the provider of the FSP service.

a) Prior to the actual provision of service, start and stop times for both the space link session and the associated FSP service instance are mutually agreed upon by SLE Complex Management and SLE Utilization Management. Configuration and other information needed to enable the service are also agreed upon.

b) Some time before the scheduled start time of the FSP service instance, the service instance is created by SLE Complex Management. Initially, the service provider is in the unbound state. At the scheduled start time of the space link session, the SLE Complexes involved establish the forward link to the spacecraft and initiate the production of FSP service and, if applicable, of the underlying FTCF and CLTU services. Typically (but not necessarily) the start time of the service instance will precede by a small margin the start time of the space link session to allow the user to bind to the service before the start of the space link session.

c) The user invokes the FSP-BIND operation to establish an association.

d) The provider transitions to the ready state and returns a report of the successful completion of the bind operation to the user.

e) The user may now check and change parameters that control service provision by means of the FSP-GET-PARAMETER and FSP-SYNC-SET-PARAMETER operations.

f) If the user is interested in obtaining periodic status reports, the FSP-SCHEDULE-STATUS-REPORT operation may be invoked to configure status reporting.

g) When the provider has determined that the production status permits the transmission of packets, it will issue a ‘production waiting’ event that informs the user that the FSP service instance is now ready to accept an FSP-START request.

h) The user now invokes the FSP-START operation to enable the submission of packets.

i) The provider transitions from the ready state to the active state and confirms the start operation to the user.

j) The user next submits a packet by invoking the FSP-TRANSFER-DATA operation. The provider verifies the TC packet, and if acceptable, buffers it until the earliest transmission start time specified in the FSP-TRANSFER-DATA operation is reached. In addition, notifications may be delivered by means of FSP-ASYNC-NOTIFY operations, and requested status reports are delivered by means of FSP-STATUS-REPORT operations.

k) The user sends additional TC packets by repeated invocation of FSP-TRANSFER-DATA operations.

l) The user checks if radiation or transfer of all TC packets is completed by inspecting the notifications or reports sent by the provider via the FSP-ASYNC-NOTIFY or FSP-STATUS-REPORT operations. The user will then invoke the FSP-STOP operation, which will cause the provider to transition to the ready state.

m) The user then may ask for the radiation of another set of frames by again invoking the FSP-START operation or terminate the association by invoking the FSP-UNBIND operation.

3 FSP SERVICE OPERATIONS

3.1 GENERAL CONSIDERATIONS

3.1.1 RESULT OF OPERATIONS

3.1.1.1 All confirmed operations shall report on the outcome of the operation in a return.

3.1.1.2 All returns except the FSP-UNBIND return shall include a `result` parameter that indicates whether the outcome of the operation was successful ('positive result') or unsuccessful ('negative result').

3.1.1.3 In the event of a 'negative result', the return shall also include a `diagnostic` parameter that is descriptive of the reason for the 'negative result'.

NOTE – Possible values of the `diagnostic` parameter are listed in the description of each operation.

3.1.1.4 A `diagnostic` of 'other' shall be returned only if no other `diagnostic` in the list adequately describes the reason for the 'negative result'.

3.1.2 PARAMETER TYPES

The types of all parameters shall conform to the abstract syntax specified in Annex A.

NOTE – Some parameter types in annex A are chosen such that possible future extensions of the range of allowed values of a parameter will not cause a type mismatch. For example, parameters that logically are of the 'enumerated' type are specified as being of the 'named integer' type.

3.1.3 PARAMETER CHECKING

3.1.3.1 Validity checks shall be performed on the parameters of operations.

3.1.3.2 If a parameter is not valid the operation shall not be performed, and, for confirmed operations other than FSP-UNBIND, a report of 'negative result' shall be returned to the invoker.

3.1.3.3 Parameter checks shall be performed in the order in which diagnostics are listed in the descriptions of the operations, and the `diagnostic` parameter shall be set to the value defined for the first problem found.

3.1.4 AUTHENTICATION

3.1.4.1 The FSP service shall include the option to authenticate all or some of the invocations and returns of FSP service operations within the application layer:

a) all invocations shall include an `invoker-credentials` parameter to permit the performer to authenticate the invocation;

b) all returns shall include a `performer-credentials` parameter to permit the invoker to authenticate the return;

c) the `invoker-credentials` or `performer-credentials` parameter may be set to the value 'unused' to signify that the invocation or return does not carry credentials.

NOTE – Requirements for security depend on the application and the SLE system environment (e.g., whether closed or public networks are used or if access is only from physically restricted areas). In many environments, security may be provided by the communications service, transparently to the SLE application.

3.1.4.2 SLE Complex Management and SLE Utilization Management shall agree on whether or not to use the credentials parameters for authentication and, if so, for which operation types.

3.1.4.3 If, by management arrangement, an invocation or return is supposed to carry credentials but does not, or the credentials it carries cannot be authenticated, then that invocation or return shall be ignored.

3.1.4.4 If, by management arrangement, an invocation or return is not supposed to carry credentials but the value of the credentials parameter is not 'unused', then that invocation or return shall be ignored.

3.1.4.5 The algorithms used to generate and authenticate credentials parameters must be mutually agreed upon by SLE Complex Management and SLE Utilization Management and must be known to both the FSP service user and provider.

NOTES

1 The specification of the algorithms themselves is outside the scope of this Recommendation.

2 This Recommendation does not preclude the use of security features that are provided by the communications service or the local environment, nor does it assume the availability of such features.

3.1.5 THREADED APPLICATIONS

3.1.5.1 To support applications that may be implemented using multiple threads of execution, the parameter `invoke-ID` shall be specified for all confirmed operations except FSP-BIND and FSP-UNBIND.

NOTE – The `invoke-ID` parameter allows the invoker to correlate a particular return to the invocation that prompted it.

3.1.5.2 The `invoke-ID` parameter shall be an arbitrary integer value provided by the invoker of an operation, which is returned, unchanged, by the performer.

3.1.5.3 An error condition shall exist if an invocation includes an `invoke-ID` whose value is the same as that of another invocation that is awaiting confirmation.

3.1.5.4 An invoker may set the value of `invoke-ID` to 'null':

a) if the invoker invokes an operation with an `invoke-ID` of 'null', the invoker shall not invoke any other operations until the return from the current operation is received;

b) if the performer receives an invocation with an `invoke-ID` of 'null', the performer shall set `invoke-ID` to 'null' in the return.

3.1.5.5 To ensure that the FSP service behaves in a predictable manner, the effects of operations shall be as though the operations were performed in the order that their invocations were received by the performer.

NOTE – This Recommendation does not assume that applications are implemented using multiple threads of execution, nor does it preclude such implementations.

3.1.6 OTHER PARAMETERS

All providers shall be able to accept and respond to parameters specified as mandatory.

NOTE – All Invoker parameters are specified as mandatory. Provider parameters are specified as conditional or mandatory. The presence of a conditional parameter is based on the value of another parameter. For some parameters, a value of null is allowed, as discussed in the description of the relevant operations.

3.1.7 TIME

Universal Time Coordinated (UTC) shall be used for all parameters containing a time value.

3.1.8 SETTING OF PARAMETERS

NOTE – Table 3-1 shows which parameters are set by service management, by FSP-START, FSP-SYNC-SET-PARAMETER, and FSP-INVOKE-DIRECTIVE.

Table 3-1: Setting of FSP Service Operation Parameters

Parameter Control	Service Management	Start	Set	Directives
tc-frame-multiplexing-scheme	X			
tc-frame-multiplexing-control	X			
map-multiplexing-scheme	X			
map-multiplexing-control				X
map-list	X			
apid-list	X			
cop-in-effect	X			
virtual-channel	X			
segmentation	X			
maximal-packet-size	X			
fecf-usage	X			
aggregation-usage (aggregation performed by the provider)	X			
maximum-segment-size	X			
maximum-frame-size	X			
transmission-limit				X
timer-initial				X
timeout-type				X
sent-queue-size (max-frame-queue-size, definition analogue CLTU)	X			
fop-sliding-window				X
fop-state	read only			
directive invocation enabled	X			
invoke-directive-counter			X	

3.1.9 ACCOUNTING SUMMARY

3.1.9.1 Statistical information to be collected over a period of time shall always refer to the service instance provision period.

3.1.9.2 Separate information shall be kept for AD and BD modes (see 1.6.1.5).

3.2 FSP-BIND

3.2.1 PURPOSE

3.2.1.1 The initiator shall invoke the FSP-BIND operation to establish an association as defined in 1.6.1.7.2.

3.2.1.2 The responder shall confirm the FSP-BIND operation.

3.2.1.3 Except as provided in 3.2.1.4, the initiator shall not invoke any further FSP operations on this service instance until the bind is confirmed.

3.2.1.4 If the return from the invocation of FSP-BIND is not received after a sufficiently long time (to be determined by service management), the initiator may attempt to recover by invoking FSP-PEER-ABORT followed by another FSP-BIND.

3.2.1.5 The FSP-BIND operation is valid only in state 1 (‘unbound’) and shall be invoked only by the user.

3.2.2 INVOCATION, RETURN, AND PARAMETERS

3.2.2.1 General

The parameters of the FSP-BIND operation shall be present in the invocation and return as specified in table 3-2.

Table 3-2: FSP-BIND Parameters

Parameter	Invocation	Return
invoker-credentials	M	
performer-credentials		M
initiator-port-identifier	M	
responder-port-identifier	M	
service-type	M	
version-number	M	C
service-instance-identifier	M	
application-context	M	C
result		M
diagnostic		C

3.2.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-BIND invocation (see 3.1.4).

3.2.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-BIND (see 3.1.4).

3.2.2.4 `initiator-port-identifier`

The **`initiator-port-identifier`** parameter shall identify the port the responder shall use to communicate with the initiator.

3.2.2.5 `responder-port-identifier`

The **`responder-port-identifier`** parameter shall identify the port to which the initiator is requesting to bind.

NOTE – The exact format and content of the `initiator-port-identifier` and `responder-port-identifier` parameters depend on the communications technology used to make the bind (see 2.6.5.4).

3.2.2.6 `service-type`

The **`service-type`** parameter shall specify the type of service that will be provided if the bind operation succeeds.[1]

3.2.2.7 `version-number`

3.2.2.7.1 The **`version-number`** parameter shall identify the version number of the FSP service specification that is to govern this association if the FSP-BIND succeeds.

3.2.2.7.2 `version-number` is conditionally included in the return based on the `result` parameter:

---

[1] For the FSP-BIND operation, the `service-type` parameter is redundant, because the only valid value of `service-type` is 'FSP'. However, it is anticipated that future work by CCSDS will result in FSP-BIND being superseded by a generic SLE-BIND operation that is invoked with any one of several SLE service types. The FSP-BIND `service-type` parameter is provided in an attempt to facilitate such a change.

a) if the value of `result` is 'positive result', `version-number` shall be included in the return;

b) if the value of `result` is 'negative result', `version-number` shall not be included.

3.2.2.7.3 If included, the responder shall either:

a) accept the version proposed by the initiator by putting the same version number into the return;

b) propose a lower (earlier) version number by placing the proposed version number in the return.

3.2.2.7.4 If the responder does not support the requested version or a lower version, it shall reject the bind with the `diagnostic`, 'version not supported'.

3.2.2.7.5 If the initiator does not support the version proposed in the return, the initiator shall unbind the association.

3.2.2.7.6 The version of the FSP service defined by this issue of this Recommendation shall be '1'.

3.2.2.8 `service-instance-identifier`

The **`service-instance-identifier`** parameter shall uniquely identify this service instance within the scope of the service-providing SLE Complex.

NOTE – It is recommended that `service-instance-identifier` take the form of a Relative Distinguished Name. Among other things, it should identify the SLE service agreement and the SLE service package applicable to this instance of service.

3.2.2.9 `application-context`

3.2.2.9.1 The **`application-context`** parameter shall specify technology-specific parameters required for association establishment and data transfer.

NOTE – The exact format and content of the `application-context` parameter depend on the communications technology used to make the bind.

3.2.2.9.2 The `application-context` parameter in the invocation shall identify a list of eligible contexts the invoker proposes to apply to the service session.

3.2.2.9.3 The `application-context` parameter is conditionally included in the return based on the `result` parameter:

a) if the value of `result` is 'positive result', `application-context` shall be included in the return;

b) if the value of `result` is 'negative result', `application-context` shall not be included.

3.2.2.9.4 If included, the responder shall either:

a) accept the application context proposed by the initiator by putting the same application context into the response;

b) propose a subset context by putting the proposed context in the return.

3.2.2.9.5 If the responder does not support the proposed context or a subset of it, it shall reject the bind with the `diagnostic` 'application context not supported'.

3.2.2.9.6 If the initiator does not support the context proposed in the return, the initiator shall unbind the association.

3.2.2.10 `result`

3.2.2.10.1 The **`result`** parameter shall specify the result of the bind operation.

3.2.2.10.2 The value of `result` shall be one of the following:

a) 'positive result'—the responder agrees to the bind request, and the association is established;

b) 'negative result'—the responder does not agree to the bind request.

3.2.2.11 `diagnostic`

3.2.2.11.1 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'no such service instance'—the requested service instance is not defined by any agreed upon service package known to the responder;

b) 'invalid time'—the request was invoked outside the transfer service instance provision period agreed to in the service package;

c) 'unable to comply'—the responder is unable to accept the bind at this time because of a fault affecting the responder;

d) 'inconsistent service type'—the `service-type` value in the request is not consistent with the service type of this service instance; i.e., it is not 'FSP';

e) 'version not supported'—the responder does not support the requested version or a lower version of the service;

f) 'application context not supported' —the requested application context is not supported by the responder;

g) 'other'—the reason for not binding will have to be found by other means.

NOTE – In some implementations, it may be inappropriate or impossible for the intended performer to provide a return in case of the conditions indicated by diagnostics a) and b). Implementations should consider that, under some conditions, FSP-BIND might fail with no return, e.g., if `responder-port` has an incorrect value.

3.2.2.11.2 If `result` is 'positive result', `diagnostic` shall not be provided.

3.2.3 EFFECTS

3.2.3.1 If the invocation is accepted:

a) the provider shall transition from state 1 ('unbound') to state 2 ('ready');

b) an association between the user and the provider shall be established;

c) upon receipt of the return with `result` set to 'positive result', the user may proceed to invoke other FSP service operations to initialize the service and enable data transfer (e.g., FSP-SYNC-SET-PARAMETER, FSP-SCHEDULE-STATUS-REPORT and FSP-START).

3.2.3.2 If the invocation is rejected:

a) the association between the user and the provider shall not be established;

b) the provider shall remain in state 1 ('unbound');

c) upon receipt of the return with `result` set to 'negative result':

1) the initiator should examine the `diagnostic` parameter for the cause;

2) the initiator may attempt to re-invoke the FSP-BIND.

3.3 FSP-UNBIND

3.3.1 PURPOSE

3.3.1.1 The initiator shall invoke the FSP-UNBIND to release an association previously established by FSP-BIND (3.2).

3.3.1.2 The responder shall confirm the FSP-UNBIND operation.

3.3.1.3 Except as provided in 3.3.1.4, the initiator shall not invoke any further FSP operations on this service instance until the unbind is confirmed.

3.3.1.4 If the return from the invocation of FSP-UNBIND is not received after a sufficiently long time (to be determined by service management), the initiator shall invoke FSP-PEER-ABORT.

3.3.1.5 The FSP-UNBIND operation is valid only in state 2 (‘ready’) and shall be invoked only by the user.

3.3.2 INVOCATION, RETURN, AND PARAMETERS

3.3.2.1 General

The parameters of the FSP-UNBIND operation shall be present in the invocation and return as specified in table 3-3.

Table 3-3: FSP-UNBIND Parameters

Parameter	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`unbind-reason`	M	

3.3.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-UNBIND invocation (see 3.1.4).

3.3.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-UNBIND (see 3.1.4).

3.3.2.4 `unbind-reason`

The **`unbind-reason`** parameter shall indicate the reason the FSP-UNBIND operation is being invoked and shall contain one of the following values:

a) 'end'—the user has completed the transfer of its data and is releasing the association normally: the provider may now release all the resources allocated to the service instance;

b) 'suspend'—the user is suspending this service usage for an unspecified period of time: the user may attempt to re-bind to the provider to continue data transfer later;

c) 'version not supported'—the user does not support the version of the FSP service proposed by the provider in the return from FSP-BIND: this value of `unbind-reason` shall be used only if the FSP-UNBIND is the first operation invoked following the FSP-BIND;

d) 'application context not supported'—the initiator does not support the application context proposed by the responder in the return from FSP-BIND; this value of `unbind-reason` shall be used only if the FSP-UNBIND is the first operation invoked following the FSP-BIND;

e) 'other'—the reason for the release will have to be found by other means.

3.3.3 EFFECTS

The FSP-UNBIND operation shall have the following effects:

a) the association between the user and the provider shall be released and the initiator and the responder shall cease to communicate with each other;

b) the provider shall transition to state 1 ('unbound');

c) if `unbind-reason` is 'end', the provider may terminate the service instance and release its resources;

d) if `unbind-reason` is not 'end', the initiator may attempt to re-bind at any time prior to the end of the service instance provision period.

NOTES

1 The responder must perform the FSP-UNBIND operation as long as it was invoked by the initiator while the provider was in state 2 ('ready'). Since there is no reason for a 'negative result', there is no result parameter in the return.

2 The performance of FSP-UNBIND for a particular service instance does not necessarily terminate FSP production since other service instances might be dependent on the production.

3.4 FSP-START

3.4.1 PURPOSE

3.4.1.1 The user shall invoke the FSP-START operation to request that the FSP service provider make ready to accept FSP-TRANSFER-DATA operations and enable the start of production (i.e., multiplexing of packets into transfer frames and delivery to the FTCF and CLTU services).

3.4.1.2 The FSP service provider shall confirm the FSP-START operation.

3.4.1.3 The FSP-START operation is valid only in state 2 ('ready') and shall be invoked only by the user.

NOTE – FSP-START is normally invoked after the user and provider have been bound by means of an FSP-BIND operation and the user has invoked any necessary FSP-SYNC-SET-PARAMETER and FSP-INVOKE-DIRECTIVE operations. This operation allows the provider to return to the user the times scheduled for start and stop of production.

3.4.2 INVOCATION, RETURN, AND PARAMETERS

3.4.2.1 General

The parameters of the FSP-START operation shall be present in the invocation and return as specified in table 3-4.

Table 3-4: FSP-START Parameters

Parameter	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`start-production-time`		M
`stop-production-time`		M
`result`		M
`diagnostic`		C

3.4.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-START invocation (see 3.1.4).

3.4.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-START (see 3.1.4).

3.4.2.4 `invoke-ID`

The FSP service provider shall return unchanged the user-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.4.2.5 `start-production-time`

3.4.2.5.1 The **`start-production-time`** parameter shall contain the estimated time that the provider will first be able to start processing buffered packets, as determined from the production schedule, with allowance for subsequent production steps and completion of the uplink acquisition sequence.

3.4.2.5.2 If there is a subsequent FSP-START operation within the same service instance, the `start-production-time` parameter shall contain the time processing first started.

3.4.2.6 `stop-production-time`

The **`stop-production-time`** parameter shall contain the estimated time at which the FSP service provider will stop processing buffered packets.

NOTE – `stop-production-time` may be later than the end of the service instance provision period.

3.4.2.7 `result`

The **`result`** parameter shall be used by the FSP service provider to specify the result of the FSP-START invocation and shall contain one of the following values:

a) ‘positive result’—the user may start issuing FSP-TRANSFER-DATA operations;

b) ‘negative result’—the provider cannot accept FSP-TRANSFER-DATA operations at this time.

3.4.2.8 `diagnostic`

3.4.2.8.1 If `result` is ‘negative result’, `diagnostic` shall be provided and shall contain one of the following values:

a) ‘duplicate Invoke-ID’—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) ‘out of service’—the provider has been taken out of service for an indefinite period by management action;

c) ‘unable to comply’—the provider is unable to accept data at this time because of a fault affecting the provider;

d) ‘production time expired’—the time period scheduled for production of packets to be passed to the forward element is already past;

e) ‘other’—the reason for rejection of the operation will have to be found by other means.

3.4.2.8.2 If `result` is ‘positive result’, `diagnostic` shall not be provided.

3.4.3 EFFECTS

3.4.3.1 If the invocation is accepted:

a) the provider shall transition to state 3 (‘active’);

b) in the ‘active’ state, the provider shall accept FSP-TRANSFER-DATA operations.

3.4.3.2 If the invocation is rejected:

a) the provider shall remain in state 2 (‘ready’);

b) Upon receipt of the return with `diagnostic` set to ‘unable to comply’,

1) the user may re-invoke the FSP-START operation at a later time;

2) if the provider’s being ‘unable to comply’ is more than a transient problem, the provider shall invoke the FSP-PEER-ABORT operation.

3.5 FSP-STOP

3.5.1 PURPOSE

3.5.1.1 The user shall invoke the FSP-STOP operation to inform the provider that the user is stopping the sending of space packets and to request the provider to stop the production of space packets already transferred by this user.

NOTE – The user may re-enable the transfer and production of packets by issuing an FSP-START operation.

3.5.1.2 The provider shall confirm the FSP-STOP operation.

3.5.1.3 The FSP-STOP operation is valid only in state 3 ('active') and shall be invoked only by the user.

3.5.2 INVOCATION, RETURN AND PARAMETERS

3.5.2.1 General

The parameters of the FSP-STOP operation shall be present in the invocation and return as specified in table 3-5.

Table 3-5: FSP-STOP Parameters

Parameters	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`result`		M
`diagnostic`		C

3.5.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-STOP invocation (see 3.1.4).

3.5.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-STOP (see 3.1.4).

3.5.2.4 `invoke-ID`

The FSP service provider shall return unchanged the user-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.5.2.5 `result`

The **`result`** parameter shall be used by the FSP service provider to specify the result of the FSP-STOP request and shall contain one of the following values:

a) 'positive result'—the user may issue an FSP-UNBIND or a new FSP-START operation;

b) 'negative result'—the provider cannot accept the FSP-STOP operation for the reason specified in `diagnostic`.

3.5.2.6 `diagnostic`

3.5.2.6.1 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'other'—the reason for rejection of the operation will have to be found by other means.

3.5.2.6.2 If `result` is 'positive result', `diagnostic` shall not be provided.

3.5.3 EFFECTS

3.5.3.1 If the invocation is accepted:

a) the FSP service provider shall transition to state 2 ('ready') and shall no longer accept FSP-TRANSFER-DATA operations;

b) for packets queued for this service instance:

   1) the provider shall discard any packets for which processing has not yet begun and send an 'end of data' notification to the user;

   2) the provider shall not discard, but shall complete processing, packets for which processing has already begun;

c) if the transmission or transfer report has been requested by the user for a packet that is already being processed, the report shall be generated and sent to the user by means of an FSP-ASYNC-NOTIFY operation.

NOTE – An FSP-ASYNC-NOTIFY operation may be invoked even though the provider has left state 3 (‘active’).

3.5.3.2 If the invocation is rejected, the provider state and the packet queue shall remain unchanged.

3.6 FSP-TRANSFER-DATA

3.6.1 PURPOSE

3.6.1.1 The FSP service user shall invoke the FSP-TRANSFER-DATA operation to transfer space packets to the FSP service provider.

3.6.1.2 The FSP service provider shall confirm the FSP-TRANSFER-DATA operation.

3.6.1.3 The FSP-TRANSFER-DATA operation is valid only in state 3 (‘active’) and shall be invoked only by the user.

3.6.2 INVOCATION, RETURN, AND PARAMETERS

3.6.2.1 General

The parameters of the FSP-TRANSFER-DATA operation shall be present in the invocation and return as specified in table 3-6.

Table 3-6: FSP-TRANSFER-DATA Parameters

Parameters	Invocation	Return
invoker-credentials	M	
performer-credentials		M
invoke-ID	M	M
earliest-production-time	M	
latest-production-time	M	
delay-time	M	
transmission-mode	M	
MAP-identifier	M	
reset-MAP	C	
aggregation	M	
radiation-report	M	
transfer-report	M	
data	M	
packet-buffer-available		M
result		M
diagnostic		C

3.6.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-TRANSFER-DATA invocation (see 3.1.4).

3.6.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-TRANSFER-DATA (see 3.1.4).

3.6.2.4 `invoke-ID`

The FSP service provider shall return unchanged the user-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.6.2.5 `earliest-production-time`

3.6.2.5.1 The **`earliest-production-time`** parameter shall be used to specify the earliest time that the provider shall start processing this packet.

3.6.2.5.2 If 'unspecified', the provider shall begin processing immediately, as long as this occurs before the production process is halted.

3.6.2.5.3 The time specified shall not be earlier than the time specified for any preceding packet.

3.6.2.6 `latest-production-time`

3.6.2.6.1 The **`latest-production-time`** parameter shall be used to specify the latest time that the provider shall start processing this packet.

3.6.2.6.2 If 'unspecified', the packet shall be processed, as long as this occurs before the production process is halted.

3.6.2.7 `delay-time`

3.6.2.7.1 The **`delay-time`** parameter shall contain the minimum radiation delay, in microseconds, between the CLTUs bearing this packet and the following CLTU.

3.6.2.7.2 If aggregation is enabled and more than one packet has been inserted into the frame that is encapsulated in the CLTU, the maximum of the `delay-time` values specified for all packets in that CLTU shall be applied.

NOTE – The `delay-time` value is passed via the FTCF service to the CLTU service. Unacceptable values will be detected by the forward CLTU service. Any notifications issued by this service will ultimately cause corresponding notifications to be issued by the forward packet service.

3.6.2.8 `transmission-mode`

The **`transmission-mode`** parameter shall specify the mode in which the packet shall be transmitted to the spacecraft and shall contain one of the following values:

a) 'sequence controlled'—the packet shall be transmitted in the sequence controlled (AD) mode;

b) 'expedited'—the packet shall be transmitted without sequence control (BD mode).

NOTE – Depending on the spacecraft telecommand decoder, it may not be possible to transmit packets in expedited mode without first explicitly terminating the AD service by means of invoking the corresponding directive.

In the case where the telecommand decoder has only a single buffer, only the user in charge of the COP protocol shall be allowed to submit BD packets. It is the user's responsibility to invoke the 'terminate AD service' directive.

3.6.2.9 `MAP-identifier`

3.6.2.9.1 The **`MAP-identifier`** parameter shall specify the Multiplexer Access Point to be used for the transmission of the space packet contained in the `data` parameter.

3.6.2.9.2 If the service instance is configured not to generate Segment Headers, this parameter must be set to 'none'.

3.6.2.10 `reset-MAP`

3.6.2.10.1 The **`reset-MAP`** parameter shall specify whether a reset-MAP command for the MAP specified in `MAP-identifier` shall be transmitted to the spacecraft prior to the packet contained in `data`.

3.6.2.10.2 The presence of the `reset-MAP` parameter is conditional on the value of `Map-identifier` and shall be absent if `MAP-identifier` is set to 'none'.

NOTE – The reset MAP command is a mechanism provided to ensure that the data management functions of the Layers above the Transfer Layer purge/reset their buffers as required for spacecraft operations. The execution of an 'Unlock' Control Command resets only FARM-1 within the Transfer Layer and not the Higher Layers. The Reset MAP command consists of a Telecommand Segment without a Segment Data Field. It includes only the Segment Header in which the Sequence Flags are set to the binary value '11' and the Multiplexer Access Point (MAP) Identifier is set to the value provided by the parameter `MAP-identifier`.

3.6.2.11 `aggregation`

3.6.2.11.1 The **`aggregation`** parameter shall specify whether the packet may be aggregated with other packets into one TC Frame.

3.6.2.11.2 If `aggregation` is set to 'yes', depending on the availability of other packets on the same MAP and the space left in the frame being built, the packet may be merged with other packets into one TC frame.

3.6.2.11.3 If `aggregation` is set to 'no', the packet shall be inserted into a frame carrying that packet only.

NOTE – Aggregation may be switched off by service management. In that case, the `aggregation` parameter will be ignored by the provider and each frame will encapsulate only one packet.

3.6.2.12 `radiation-report`

The **`radiation-report`** parameter shall specify whether the provider shall issue an FSP-ASYNC-NOTIFY, upon completion of the radiation of the packet, and shall contain one of the following values:

a) 'produce report'—invoke an FSP-ASYNC-NOTIFY operation upon completion of the radiation;

b) 'do not produce report'—do not invoke the FSP-ASYNC-NOTIFY operation.

3.6.2.13 `transfer-report`

The **`transfer-report`** parameter shall specify whether the provider shall issue an FSP-ASYNC-NOTIFY upon completion of the transfer of the packet, i.e., when all segments of the packet have been acknowledged by the spacecraft in the CLCW, and shall contain one of the following values:

a) ‘produce report’—invoke an FSP-ASYNC-NOTIFY operation upon completion of the transfer. This value is allowed only when `transmission-mode` is ‘sequence controlled’;

b) ‘do not produce report’—do not invoke the FSP-ASYNC-NOTIFY operation.

3.6.2.14 `data`

The **`data`** parameter shall contain a telecommand packet for transfer to the space element of a mission.

3.6.2.15 `packet-buffer-available`

The **`packet-buffer-available`** parameter shall specify the remaining number of octets available to this service instance for buffering of space packets.

3.6.2.16 `result`

The **`result`** parameter shall specify the result of the FSP-TRANSFER-DATA invocation and shall contain one of the following values:

a) ‘positive result’—the FSP-TRANSFER-DATA invocation has been checked successfully, and the data will be queued by the provider until processing is due;

b) ‘negative result’—the FSP-TRANSFER-DATA invocation has failed an error check, for the reason returned in the `diagnostic` parameter.

3.6.2.17 `diagnostic`

3.6.2.17.1 If `result` is ‘negative result’, `diagnostic` shall be provided and shall contain on one of the following values:

a) ‘duplicate Invoke-ID’—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) ‘unable to process’—the provider cannot process packets due to a halted, interrupted, or expired exception condition in production (see 3.7);

c) ‘invalid transmission mode’—a transmission mode has been invoked which is not allowed for that particular service instance or currently not configured for the VC of this service instance;

d) ‘inconsistent production time range’—the specified earliest-production-time is later than the specified latest-production-time;

e) ‘invalid production time’—the interval specified during which radiation would have to occur does not fall within the time for which the service instance is scheduled;

f) ‘late sldu’—the latest-production-time is earlier than the time the FSP-TRANSFER-DATA operation is received by the provider;

g) ‘packet too long’—the packet contained in data exceeds the maximum packet size allowed for this service instance or the packet requires segmentation while segmentation is disabled;

h) ‘unsupported packet version’—the packet version is either not supported by CCSDS or is supported by CCSDS but not specified in the service package;

i) ‘incorrect packet type’—the packet type does not specify telecommand;

j) ‘invalid packet apid’—the AP-ID in the packet header is not one specified for this service instance;

k) ‘invalid MAP’—the value in the map-identifier parameter is not one specified for this service instance or a MAP is specified although no Segment Headers should be generated;

l) ‘unable to store’—there is not sufficient buffer space available to store this packet;

m) ‘other’—the reason for rejection of the operation will have to be found by other means.

3.6.2.17.2 If `result` is ‘positive result’, `diagnostic` shall not be provided.

3.6.3 EFFECTS

If the invocation is accepted:

a) the provider shall queue the packet until it is due to be processed;

b) packets shall be processed as appended to the tail of the queue and according to the production time parameters;

c) the service shall remain in state 3 (‘active’).

NOTE – A `result` of ‘negative result’ in the return indicates that the invocation was rejected and that the packet was not buffered.

3.7 FSP-ASYNC-NOTIFY

3.7.1 PURPOSE

3.7.1.1 The FSP service provider shall invoke the FSP-ASYNC-NOTIFY operation to notify the user of an event affecting the production of the FSP-service.

3.7.1.2 The FSP service user shall confirm the FSP-ASYNC-NOTIFY invocation.

3.7.1.3 The FSP-ASYNC-NOTIFY operation is valid only in states 3 ('active') and 2 ('ready') and shall be invoked only by the provider.

3.7.2 INVOCATION, RETURN, AND PARAMETERS

3.7.2.1 General

The parameters of the FSP-ASYNC-NOTIFY operation shall be present in the invocation and return as specified in table 3-7.

Table 3-7: FSP-ASYNC-NOTIFY Parameters

Parameters	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`notification-type`	M	
`set-executed-identifier`	M	
`directive-executed-identification`	M	
`event-thrown-identifier`	M	
`packet-header-last-ok`	M	
`packet-header-last-processed`	M	
`production-start-time`	M	
`production-stop-time`	M	
`packet-status`	M	
`production-status`	M	
`result`		M
`diagnostic`		C

3.7.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-ASYNC-NOTIFY invocation (see 3.1.4).

3.7.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-ASYNC-NOTIFY (see 3.1.4).

3.7.2.4 `invoke-ID`

The performer shall return unchanged the invoker-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.7.2.5 `notification-type`

The **`notification-type`** parameter shall describe the event being notified to the user and shall contain one of the following values:

a) 'packet radiated'—the packet whose header is contained in the `packet-header-last-processed` parameter completed radiation without abortion. For sequence controlled packet transmission this may imply that at least one acknowledgement via a CLCW is still outstanding.

b) 'packet transferred'—the packet whose header is contained in the `packet-header-last-processed` parameter completed radiation without abortion. This applies only to sequence controlled transmission mode and signifies that all packet components have been acknowledged by the space element via the associated stream of CLCWs.

NOTE – The packet is guaranteed to have been *received* by the space element, but not necessarily *accepted* by the space segment. Packet assembly or execution may not have been carried out correctly. This can only be determined by examining telemetry.

c) 'sldu expired'—processing did not begin by the time specified in the `latest-production-time` parameter for the reported `last-processed-packet`. The production process is stopped for packets belonging to the affected service instance and further FSP-TRANSFER-DATA invocations will be rejected.

NOTE – After the 'sldu expired' event, in order to resume the transfer of packets, the service user has to clear this event condition by first invoking an FSP-STOP and then an FSP-START (after confirmation of the FSP-STOP operation).

d) 'packet transmission mode mismatch'—the provider configuration does not permit the transmission of a packet in the requested mode. The requested mode may be disabled by service management or the provider configuration resulting from FSP-INVOKE-DIRECTIVE invocations for the moment prevents the transmission of packets in the requested mode.

NOTE – After the 'packet transmission mode mismatch' event, the production of packets of the affected service instance is stopped, all packets buffered for this instance are discarded and further FSP-TRANSFER-DATA invocations will be rejected. In order to resume the transfer of data, the service user has to clear this event condition by first invoking an FSP-STOP and then an FSP-START (after confirmation of the FSP-STOP operation).

e) 'production interrupted'—processing aborted for the reported `last-processed-packet`. The production process is stopped; however, it may still be functional.

NOTE – After the 'production interrupted' event, the production process is stopped, all buffered packets are discarded and further FSP-TRANSFER-DATA invocations will be rejected. In order to resume the transfer of data, the service user must first invoke an FSP-STOP and wait for a 'production waiting' event. After receiving this event, the service user may then invoke an FSP-START.

f) 'production halted'—this event occurs when the production process is unable to continue. The halted state is entered by service management decision, for example in case of malfunctioning equipment.

NOTE – If the production process is stopped, all buffered packets are discarded and further FSP-TRANSFER-DATA invocations will be rejected. In order to resume the transfer of data, the service user must first invoke an FSP-STOP and wait for a 'production waiting' event. After receiving this event, the service user may then invoke an FSP-START.

g) 'production waiting'—the production process is ready. This event occurs when the uplink has been successfully established or following actions to restore production to ready status after a halt or interrupt.

h) 'no invoke directive capability on this VC'—the service instance for which the FSP-INVOKE-DIRECTIVE is enabled is no longer connected to the service provider because of an FSP-UNBIND, FSP-PEER-ABORT, or FSP-PROTOCOL-ABORT for that instance. This notification is not applicable to the service instance that has been enabled to invoke the FSP-INVOKE-DIRECTIVE operation.

i) 'end of data'—the provider has no further packets buffered for this service instance. Note that this event may be generated as the result of an FSP-STOP operation.

j) 'positive confirm response to directive'—the directive identified in the parameter `directive-identification-executed` completed successfully. This notification indicates that COP-1 (including FARM-1 for directives requiring receiving-end action) was able to complete the execution of the Directive.

k) 'negative confirm response to directive'—the directive identified in the parameter `directive-identification-executed` did not complete successfully. The 'negative confirm response to directive' notification does not carry a parameter giving

the reason for the failure to confirm performance of the actions invoked by the Directive. However, whenever a condition is detected which might give rise to 'Negative Confirm Response to Directive' notification, an 'Alert' notification shall be sent by the provider.

l) 'parameter modified'—the FSP parameter has been updated as requested via the FSP-SYNC-SET-PARAMETER operation identified in the `set-executed-identifier` parameter.

m) 'suspend'—the allowed number of transmissions has been completed and the service has been suspended. A subsequent 'Resume' Directive will then cause FOP-1 to resume service in the same state it was in when it was suspended.

n) 'action list completed'—all actions associated with the event triggered by an FSP-THROW-EVENT operation invoked earlier by the user and identified in the parameter `event-invocation-identifier` are successfully completed.

o) 'action list not completed'—at least one of the actions of the action list associated with the event triggered by the FSP-THROW-EVENT operation invoked earlier by the user and identified in the parameter `event-invocation-identifier` did not complete.

p) 'event condition evaluated to false'—the event condition specified for the event attempted to be triggered by the FSP-THROW-EVENT operation invoked earlier by the user and identified in the parameter `event-invocation-identifier` evaluated to false. Therefore, the actions associated with this event have not been performed.

q) 'FOP alert'—this alert is generated by the FSP production as notification of the termination of the sequence-controlled service guarantee. It is one of the following:

1) 'FOP alert limit'—allowed number of transmissions exhausted for a Type-AD frame or the allowed number of transmissions exhausted for a Type-BC frame derived from a directive (e.g., 'Initiate AD Service' Directive with 'Unlock' or with 'Set V(R)');

2) 'FOP alert-lockout'—lockout detected;

3) 'FOP alert-synch'—CLCW with 'Retransmit' Flag = 0 and N(R) = NN(R) has arrived, when last CLCW showed 'Retransmit' Flag = 1 or all frames sent are acknowledged but 'Retransmit' Flag = 1 or an attempt to acknowledge frames is made during the initializing phase corresponding to FOP State (S4);

4) 'FOP alert-NN(R)'—CLCW with invalid N(R) has arrived;

5) 'FOP alert-CLCW'—CLCW with = 'Wait' Flag = 1 and 'Retransmit' Flag = 0 has arrived or CLCW with invalid pattern of bits has arrived;

6) ‘FOP alert-LLIF’—FOP-1 and Lower Layer are out of synchronization;

7) ‘FOP alert-term’—a ‘Terminate AD Service’ Directive has arrived.

3.7.2.6 `set-executed-identifier`

3.7.2.6.1 If `notification-type` is ‘parameter modified’, the **`set-executed-identifier`** parameter shall indicate the `set-identifier` value of the FSP-SYNC-SET-PARAMETER invocation that resulted in the change of parameter reported in the notification.

3.7.2.6.2 For all other notifications, this parameter shall be set to ‘null’.

3.7.2.7 `directive-executed-identification`

3.7.2.7.1 If `notification-type` is ‘positive confirm response to directive’ or ‘negative confirm response to directive’, the **`directive-executed-identification`** parameter shall identify the `directive-identification` value in the FSP-INVOKE-DIRECTIVE invocation.

3.7.2.7.2 For all other notifications, this parameter shall be set to ‘null’.

3.7.2.8 `event-thrown-identifier`

3.7.2.8.1 If `notification-type` is ‘action list completed’, ‘action list not completed’, or ‘event condition evaluate to false’, the **`event-thrown-identifier`** parameter shall indicate the `event-invocation-identifier` value of the FSP-THROW-EVENT invocation that triggered the actions whose results are reported.

3.7.2.8.2 For all other notifications, this parameter shall be set to ‘null’.

3.7.2.9 `packet-header-last-ok`

3.7.2.9.1 The **`packet-header-last-ok`** parameter shall identify the last successfully processed packet by returning this packet’s header.

3.7.2.9.2 If no packets have been processed during this service instance, this parameter shall be set to ‘null’.

3.7.2.10 `packet-header-last-processed`

3.7.2.10.1 The **`packet-header-last-processed`** parameter shall identify the packet most recently processed by returning this packet’s header.

3.7.2.10.2 If no packets have been processed during this service instance, this parameter shall be set to 'null'.

NOTE – The most recently processed packet may be the same as that identified in `packet-header-last-ok`.

3.7.2.11 `production-start-time`

3.7.2.11.1 The **`production-start-time`** parameter shall contain the UTC time at which the provider started to process the packet identified in `packet-header-last-processed`.

3.7.2.11.2 If no packets have been processed during this service instance, this parameter shall be set to 'null'.

3.7.2.12 `production-stop-time`

3.7.2.12.1 For expedited mode, the **`production-stop-time`** parameter shall contain the UTC time at which transmission of the packet identified by `packet-header-last-processed` was accomplished.

3.7.2.12.2 For sequence controlled transmission mode, it shall contain the UTC time at which the last packet component was acknowledged via a received CLCW.

3.7.2.12.3 In the case of abnormal completion, this parameter shall contain the time when the abort occurred.

3.7.2.12.4 If no packets have been processed during this service instance, or the packet reported on is currently being processed, this parameter shall be set to 'null'.

3.7.2.13 `packet-status`

The **`packet-status`** parameter shall represent the processing state of the packet identified by `packet-header-last-processed` and shall contain one of the following values:

a) 'transferred'—(applies only to sequence controlled transmission mode) all packet components have been acknowledged by the space element via the associated stream of CLCWs.

NOTE – If `packet-status` is 'transferred', the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed to have been *received* by the space element, but not necessarily *accepted* by the space segment. Packet assembly or execution may not have been carried out correctly. This can only be determined by examining telemetry.

b) ‘radiated’—the radiation of the packet components completed without aborting. For sequence controlled packet transmission this may imply that at least one acknowledgement via a CLCW is still outstanding.

NOTE – If the `packet-status` is ‘radiated’, the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed to have been radiated nominally.

c) ‘production started’—the packet has been removed from the service instance's request queue (see annex B) and that processing has commenced.

d) ‘expired’—the Latest-production-time was reached before beginning processing.

NOTE – If the `packet-status` is ‘expired’, the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed not to have been radiated.

e) ‘interrupted’—processing was aborted before reaching nominal completion.

NOTE – If the `packet-status` is ‘interrupted’, the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed to have started being processed, and to have aborted prior to nominal completion. Earlier packets may also have been affected.

f) ‘undefined’—there is no packet to be reported on.

3.7.2.14 `production-status`

The **`production-status`** parameter shall describe the current status of the production process and shall contain one of the following values:

a) ‘waiting’—the production process is capable of, but is not currently processing packets. This status occurs under the following conditions:

   1) when the production equipment is configured for support and has completed the acquisition sequence but before a successful FSP-START;

   2) once processing has stopped after an FSP-STOP;

   3) once processing has stopped after an FSP-ASYNC-NOTIFY, unless service management has caused a production halt (in this case the status is ‘halted’).

b) ‘running’—the production process is capable of, and is currently processing packets if available. This state is entered after a successful FSP-START following production equipment configuration and completion of the acquisition sequence.

c) ‘halted’—the production process is inactive and unable to process packets until action is taken by service management.

3.7.2.15 `result`

The **`result`** parameter shall specify the result of the FSP-ASYNC-NOTIFY invocation and shall contain one of the following values:

a) 'positive result'—the user has accepted the notification;

b) 'negative result'—the user did not accept the notification.

3.7.2.16 `diagnostic`

3.7.2.16.1 The **`diagnostic`** parameter is conditional on, and shall be used to clarify, a 'negative result'.

3.7.2.16.2 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'other'—the reason for rejection of the operation will have to be found by other means.

3.7.3 EFFECTS

If `notification` indicates 'sldu expired', 'packet transmission mode mismatch', 'production interrupted' or 'production halted':

a) the production process shall be stopped;

b) all packets queued for this service instance shall be discarded;

c) further FSP-TRANSFER-DATA invocations shall be rejected.

3.8 FSP-SCHEDULE-STATUS-REPORT

3.8.1 PURPOSE

3.8.1.1 The user shall invoke the FSP-SCHEDULE-STATUS-REPORT operation to request that the provider do one of the following:

a) send a status report immediately;

b) send a status report periodically;

c) stop sending periodic status reports.

3.8.1.2 The provider shall confirm the FSP-SCHEDULE-STATUS-REPORT invocation.

3.8.1.3 The provider shall deliver the requested report(s) by means of the FSP-STATUS-REPORT operation (see 3.9).

3.8.1.4 For periodic reporting, the user may change the reporting period by submitting another FSP-SCHEDULE-STATUS-REPORT operation.

3.8.1.5 The FSP-SCHEDULE-STATUS-REPORT operation is valid only in states 2 (‘ready’) and 3 (‘active’) and shall be invoked only by the user.

3.8.2 INVOCATION, RETURN, AND PARAMETERS

3.8.2.1 General

The parameters of the FSP-SCHEDULE-STATUS-REPORT operation shall be present in the invocation and return as specified in table 3-8.

Table 3-8: FSP-SCHEDULE-STATUS-REPORT Parameters

Parameters	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`report-request-type`	M	
`reporting-cycle`	C	
`result`		M
`diagnostic`		C

3.8.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-SCHEDULE-STATUS-REPORT invocation (see 3.1.4).

3.8.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-SCHEDULE-STATUS-REPORT (see 3.1.4).

3.8.2.4 `invoke-ID`

The performer shall return unchanged the invoker-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.8.2.5 `report-request-type`

3.8.2.5.1 The parameter **`report-request-type`** shall specify how reporting shall be done and shall contain one of the following values:

a) 'immediately'—send a single status report immediately;

b) 'periodically'—send a status report every `reporting-cycle` value seconds;

c) 'stop'—do not send further status reports.

3.8.2.5.2 If `report-request-type` is 'immediately',

a) the provider shall stop sending status reports after the immediate status report has been sent;

b) periodic reporting may be restarted by means of another FSP-SCHEDULE-STATUS-REPORT invocation.

3.8.2.6 `reporting-cycle`

3.8.2.6.1 If `report-request-type` is 'periodically', then the `reporting-cycle` parameter shall be present and shall specify the requested interval between status reports in seconds.

3.8.2.6.2 If `report-request-type` parameter is not 'periodically', the `reporting-cycle` parameter shall be absent.

3.8.2.7 `result`

The **`result`** parameter shall specify the result of the FSP-SCHEDULE-STATUS-REPORT invocation and shall contain one of the following values:

a) 'positive result'—the request for status report is accepted, and the provider will send the requested report(s);

b) 'negative result'—the request for status reports is rejected, and the previous setting for status reporting remains in effect.

3.8.2.8 `diagnostic`

3.8.2.8.1 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'already stopped'—the provider is not currently providing periodic reports (when `report-request-type is` 'stop');

c) 'invalid reporting cycle'—the `reporting-cycle` value is outside the range agreed upon by management arrangement;

d) 'other'—the reason for rejection of the operation will have to be found by other means.

3.8.2.8.2 If `result` is 'positive result', `diagnostic` shall not be provided.

3.8.3 EFFECTS

3.8.3.1 The service shall remain in its original state, i.e., state 2 ('ready') or state 3 ('active').

3.8.3.2 If the invocation is accepted:

a) if `report-request-type` specifies 'immediately', a status report shall be sent immediately;

b) if `report-request-type` specifies 'periodically', the first FSP-STATUS-REPORT shall be sent immediately and subsequent reports at the interval specified in `reporting-cycle`;

c) if `report-request-type` specifies 'stop', periodic reporting shall cease.

3.8.3.3 If the invocation is rejected, there shall be no change to the current reporting.

3.9 FSP-STATUS-REPORT

3.9.1 PURPOSE

3.9.1.1 The provider shall invoke the FSP-STATUS-REPORT operation to send a status report to the user.

3.9.1.2 Status reports shall be sent (or not sent) in accordance with the user requests conveyed by means of the FSP-SCHEDULE-STATUS-REPORT operation (see 3.8).

3.9.1.3 The FSP-STATUS-REPORT operation is valid only in states 2 (‘ready’) and 3 (‘active’) and shall be invoked only by the provider.

NOTE – FSP-STATUS-REPORT is an unconfirmed operation.

3.9.2 INVOCATION AND PARAMETERS

3.9.2.1 General

The parameters of the FSP-STATUS-REPORT operation shall be present in the invocation and return as specified in table 3-9.

Table 3-9: FSP-STATUS-REPORT Parameters

Parameters	Invocation
invoker-credentials	M
packet-header-last-ok	M
packet-header-last-processed	M
production-start-time	M
production-stop-time	M
report-packet-status	M
production-status	M
number-of-packets-received	M
number-of-packets-processed	M
number-of-packets-radiated	M
number-of-packets-transferred	M
packet-buffer-available	M

3.9.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-STATUS-REPORT invocation (see 3.1.4).

3.9.2.3 `packet-header-last-ok`

3.9.2.3.1 The **`packet-header-last-ok`** parameter shall identify the last successfully processed packet by returning this packet's header.

3.9.2.3.2 If no packets have been processed so far on behalf of the user of the given service instance, this parameter shall be set to 'null'.

3.9.2.4 `packet-header-last-processed`

3.9.2.4.1 The **`packet-header-last-processed`** parameter shall identify the packet most recently processed by returning this packet's header.

3.9.2.4.2 If no packets have been processed so far on behalf of the user of the given service instance, this parameter shall be set to 'null'.

NOTE – The `packet-header-last-processed` value may be identical to the value of `packet-header-last-ok`.

3.9.2.5 `production-start-time`

3.9.2.5.1 The **`production-start-time`** parameter shall contain the UTC time at which the provider started to process the packet identified in `packet-header-last-processed`.

3.9.2.5.2 If no packets have been processed during this service instance, this parameter shall be set to 'null'.

3.9.2.6 `production-stop-time`

3.9.2.6.1 The parameter **`production-stop-time`** for normal and bypass transmission mode shall contain the UTC time at which transmission of the packet identified by `packet-header-last-processed` was accomplished.

3.9.2.6.2 For a sequence controlled transmission mode, it shall contain the UTC time at which the last packet component was acknowledged via a received CLCW.

3.9.2.6.3 In the case of abnormal completion, it shall contain the UTC time when the abort occurred.

3.9.2.6.4 If no packets have been processed during this service instance, or the packet reported on is currently being processed, this parameter shall be set to 'null'.

3.9.2.7 `report-packet-status`

The **`report-packet-status`** parameter shall represent the processing state of the packet identified by `packet-header-last-processed` and shall contain one of the following values:

a) 'transferred'—(applies only to sequence controlled transmission mode) all packet components have been acknowledged by the space element via the associated stream of CLCWs.

NOTE – If the `packet-status` is 'transferred', the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed to have been *received* by the space element, but not necessarily *accepted* by the space segment. Packet assembly or execution may not have been carried out correctly. This can only be determined by examining telemetry.

b) 'radiated'—the radiation of the packet components completed without aborting. For sequence controlled packet transmission this may imply that at least one acknowledgement via a CLCW is still outstanding.

NOTE – If the `packet-status` is 'radiated', the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed to have been radiated nominally.

c) 'production started'—the packet has been removed from the service instance queue and processing has commenced.

d) 'expired'—the Latest-production-time was reached before beginning processing.

NOTE – If the `packet-status` is 'expired', the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed not to have been radiated.

e) 'interrupted'—processing was aborted before reaching nominal completion.

NOTE – If the `packet-status` is 'interrupted', the packet whose header is returned in the FSP-ASYNC-NOTIFY is guaranteed to have started being processed, and to have aborted prior to nominal completion. Earlier packets may also have been affected.

f) 'undefined'—there is no packet to be reported on.

3.9.2.8 `production-status`

The **`production-status`** parameter shall describe the status of the production process and shall contain one of the following values:

a) 'waiting'—the production process is capable of, but is not currently processing packets. This status occurs under the following conditions:

   1) when the production equipment is configured for support and has completed the acquisition sequence but before a successful FSP-START;

   2) once processing has stopped after an FSP-STOP;

   3) once processing has stopped after an FSP-ASYNC-NOTIFY, unless service management has caused a production halt (in which case the status is 'halted').

b) 'running'—the production process is capable of, and is currently processing packets if available. This state is entered after a successful FSP-START following production equipment configuration and completion of the acquisition sequence.

c) 'halted'—the production process is inactive and unable to process packets until action is taken by service management.

3.9.2.9 `number-of-packets-received`

3.9.2.9.1 The **`number-of-packets-received`** parameter shall specify the number of packets received during this service instance.

3.9.2.9.2 Only packets that are checked successfully and buffered shall be counted in the total.

3.9.2.9.3 This parameter shall include separate values for AD packets and BD packets.

3.9.2.10 `number-of-packets-processed`

3.9.2.10.1 The **`number-of-packets-processed`** parameter shall specify the number of packets that the provider attempted to process during this service instance.

3.9.2.10.2 Packets that are determined to have expired shall be included in the count.

3.9.2.10.3 Any packets currently being processed shall be included in this count as well.

3.9.2.10.4 If no packets have been processed, this parameter shall be set to 'null'.

3.9.2.10.5 This parameter shall include separate values for AD packets and BD packets.

3.9.2.11 `number-of-packets-radiated`

3.9.2.11.1 The **`number-of-packets-radiated`** parameter shall specify the number of packets successfully radiated during this service instance.

3.9.2.11.2 A packet in the process of being radiated shall not be included in this count.

3.9.2.11.3 For a given packet, only its first radiation shall be counted; retransmission of frames in accordance with COP-1, which may imply retransmission of complete packets, shall not cause this parameter to be updated.

3.9.2.11.4 If no packets have been radiated, this parameter shall be set to 'null'.

3.9.2.11.5 This parameter shall include separate values for AD packets and BD packets.

3.9.2.12 `number-of-packets-transferred`

3.9.2.12.1 The **`number-of-packets-transferred`** parameter only applies to sequence controlled transmission mode and shall specify the number of packets that arrived in the space element during this service instance.

3.9.2.12.2 If no packets have been transferred, this parameter shall be set to 'null'.

3.9.2.13 `packet-buffer-available`

The **`packet-buffer-available`** parameter shall specify the number of octets in the buffer available to this service instance for packet storage.

3.9.3 EFFECTS

3.9.3.1 Status information shall be delivered to the user.

3.9.3.2 The state of the provider shall remain unchanged.

3.10 FSP-SYNC-SET-PARAMETER

3.10.1 PURPOSE

3.10.1.1 The user shall invoke the FSP-SYNC-SET-PARAMETER operation to set an FSP parameter that controls the service provision.

3.10.1.2 The provider shall confirm the FSP-SYNC-SET-PARAMETER invocation.

NOTE – The operation is buffered. The return will only show whether the operation itself has been accepted or rejected.

3.10.1.3 The provider shall change the specified parameter to the requested valued and inform the user of the change via an FSP-ASYNC-NOTIFY operation (see 3.7).

3.10.1.4 The FSP-SYNC-SET-PARAMETER operation is valid only in states 2 (‘ready’) and 3 (‘active’) and shall be invoked only by the user.

3.10.2 INVOCATION, RETURN, AND PARAMETERS

3.10.2.1 General

The parameters of the FSP-SYNC-SET-PARAMETER operation shall be present in the invocation and return as specified in table 3-10.

Table 3-10: FSP-SYNC-SET-PARAMETER Parameters

Parameters	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`fsp-parameter`	M	
`parameter-value`	M	
`set-identifier`	M	M
`result`		M
`diagnostic`		C

3.10.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-SYNC-SET-PARAMETER invocation (see 3.1.4).

3.10.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-SYNC-SET-PARAMETER (see 3.1.4).

3.10.2.4 `invoke-ID`

The performer shall return unchanged the invoker-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.10.2.5 `fsp-parameter`

The **`fsp-parameter`** parameter shall specify the parameter to set and shall contain one of the values listed in table 3-11.

Table 3-11: `fsp-parameter` Parameter

Parameter	Description
'directive-identification'	Specifies the value that the provider shall expect in the next FSP-INVOKE-DIRECTIVE operation. This parameter may only be set by the COP user.

3.10.2.6 `parameter-value`

3.10.2.6.1 The **`parameter-value`** parameter shall specify the value for the parameter specified in the `fsp-parameter` parameter.

3.10.2.6.2 Its value is dependent on which parameter was selected and shall be constrained to a range of values specified in the service package in force.

3.10.2.7 `set-identifier`

3.10.2.7.1 The **`set-identifier`** parameter shall contain a monotonically increasing sequence number and shall be

a) set to '0' for the first FSP-SYNC-SET-PARAMETER invocation following the first successful FSP-BIND to the service instance;

b) incremented for each FSP-SYNC-SET-PARAMETER invocation accepted by the provider.

3.10.2.7.2 The provider shall return the value expected in the next FSP-SYNC-SET-PARAMETER operation:

a) if the invocation is accepted, the value of `set-identifier` in the return shall be one greater than the value of the invocation parameter;

b) if the invocation is rejected, `set-identifier` in the return shall contain the value expected by the provider.

3.10.2.7.3 The provider shall use the user-supplied value of `set-identifier` to reference this invocation in subsequent notifications (FSP-ASYNC-NOTIFY).

3.10.2.7.4 The value expected by the provider shall not be altered when the association between user and provider is reestablished after being released or aborted.

3.10.2.8 `result`

The **`result`** parameter shall specify the result of the FSP-SYNC-SET-PARAMETER invocation and shall contain one of the following values:

a) 'positive result'—the provider has queued the request;

b) 'negative result'—the provider did not accept the invocation.

3.10.2.9 `diagnostic`

If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'unknown parameter'—the parameter identified by `fsp-parameter` is not known to the service provider;

c) 'invalid parameter'—the parameter identified by `fsp-parameter` may not be set by the user through this FSP-SYNC-SET-PARAMETER operation; e.g., the Maximum_Packet_Size is 'read only' for the FSP user;

d) 'value out of range'—the value of `parameter-value` is outside the range allowed for the parameter identified by `fsp-parameter`;

e) 'other'—the reason for rejection of the operation will have to be found by other means.

3.10.3 EFFECTS

3.10.3.1 If the invocation is accepted, the operation shall be buffered until all operations previously invoked by the user and accepted by the provider have completed.

3.10.3.2 The provider shall then change the parameter setting to be consistent with the requested parameter value.

3.10.3.3 Upon completion of this parameter change, the provider shall notify the user by invoking FSP-ASYNC-NOTIFY.

3.10.3.4 The service shall remain in its original state, i.e., state 2 (‘ready’) or state 3 (‘active’).

3.11 FSP-GET-PARAMETER

3.11.1 PURPOSE

3.11.1.1 The user shall invoke the FSP-GET-PARAMETER operation to ascertain the value of an FSP service parameter.

3.11.1.2 The provider shall confirm the FSP-GET-PARAMETER invocation and, if `result` is 'positive-result', return the current value of the specified FSP parameter.

3.11.1.3 The FSP-GET-PARAMETER operation is valid only in states 2 ('ready') and 3 ('active') and shall be invoked only by the user.

3.11.2 INVOCATION, RETURN, AND PARAMETERS

3.11.2.1 General

The parameters of the FSP-GET-PARAMETER operation shall be present in the invocation and return as specified in table 3-12.

Table 3-12: FSP-GET-PARAMETER Parameters

Parameters	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`fsp-parameter`	M	M
`parameter-value`		M
`result`		M
`diagnostic`		C

3.11.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-GET-PARAMETER invocation (see 3.1.4).

3.11.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-GET-PARAMETER (see 3.1.4).

3.11.2.4 `invoke-ID`

The performer shall return unchanged the invoker-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.11.2.5 `fsp-parameter`

The **`fsp-parameter`** parameter shall specify which parameter to get and shall contain one of the values listed in table 3-13.

Table 3-13: FSP Parameters

Parameter	Description
vc-multiplexing-scheme	Identifies the VC multiplexing scheme in effect ('FIFO', 'absolute priority', 'polling vector').
vc-multiplexing-control	Identifies the VC priority list or the VC polling vector; if the vc-multiplexing scheme is 'FIFO', this parameter is 'null'.
map-multiplexing-scheme	Identifies the MAP multiplexing scheme in effect ('FIFO', 'absolute priority', 'polling vector').
map-multiplexing-control	Identifies the VC priority list or the VC polling vector; if the vc-multiplexing scheme is 'FIFO', this parameter is 'null'.
directive-identification	Identifies the current value of a particular directive invocation identifier.
cop-in-effect	Identifies Command Operation Procedure currently in use.
virtual-channel	Identifies Virtual Channel being used by this Service Instance.
Segmentation	'segment header present' if segmentation is required and/or MAPs are used, 'no segmentation applied' if the segment header shall be absent.
maximum-packet-size	Mission maximum telecommand packet size in octets.
fecf-usage	'used' if frame error control used, 'not used' otherwise.
Aggregation-usage	'permitted' if packets may be aggregated in a frame, 'not permitted' otherwise.
maximum-segment-size	Mission maximum segment size in octets.
maximum-frame-size	Mission maximum frame size in octets.
Transmission-limit	Maximum number of times first frame on Sent_Queue may be transmitted.
timer-initial	Initial value for countdown timer when an AD or BC frame is transmitted.
timeout-type	Timeout type. Specifies either 'Alert' or 'AD service suspension' whenever the countdown timer expires and the transmission limit is exceeded.
sent-queue-size	Size of queue used for holding AD and BC frames until processed by the Transfer Layer in octets.
fop-sliding-window	Number of frames that can be transmitted before an acknowledgement is required.
fop-state	'ACTIVE', 'RETRANSMIT WITHOUT WAIT', 'RETRANSMIT WITH WAIT', 'INITIALIZING WITHOUT BC FRAME', 'INITIALIZING WITH BC FRAME', 'INITIAL'

3.11.2.6 `parameter-value`

The **`parameter-value`** parameter shall contain the value for the parameter specified in the `fsp-parameter` parameter.

3.11.2.7 `result`

The **`result`** parameter shall specify the result of the FSP-GET-PARAMETER invocation and shall contain one of the following values:

a) 'positive result'—the provider retrieved the requested parameter;

b) 'negative result'—the provider did not retrieve the requested parameter.

3.11.2.8 `diagnostic`

3.11.2.8.1 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'unknown parameter'—the `fsp-parameter` specified is not known to the service provider;

c) 'other'—the reason for rejection of the operation will have to be found by other means.

3.11.2.8.2 If `result` is 'positive result', `diagnostic` shall not be provided.

3.11.3 EFFECTS

If the invocation is accepted:

a) the value of the specified parameter shall be communicated to the user;

b) the provider state shall remain unchanged.

3.12 FSP-THROW-EVENT

3.12.1 PURPOSE

3.12.1.1 The user shall invoke the FSP-THROW-EVENT operation to forward an event that requires management action to Complex Management and, where applicable, to underlying SLE service (FTCF, CLTU) providers.

NOTE – An example of such an event is change of the uplink bit rate. Event handlers are described in [F4].

3.12.1.2 The performer shall confirm the FSP-THROW-EVENT invocation.

3.12.1.3 On the completion of the action list associated with the event, the performer shall inform the user via an FSP-ASYNC-NOTIFY operation.

3.12.1.4 The FSP-THROW-EVENT operation is valid only in states 2 (‘ready’) and 3 (‘active’) and shall be invoked only by the user.

3.12.2 INVOCATION, RETURN, AND PARAMETERS

3.12.2.1 General

The parameters of the FSP-THROW-EVENT operation shall be present in the invocation and return as specified in table 3-14.

Table 3-14: FSP-THROW-EVENT Parameters

Parameters	Invocation	Return
`invoker-credentials`	M	
`performer-credentials`		M
`invoke-ID`	M	M
`event-identification`	M	
`event-list-version`	M	
`event-invocation-identifier`	M	M
`result`		M
`diagnostic`		C

3.12.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-THROW-EVENT invocation (see 3.1.4).

3.12.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-THROW-EVENT (see 3.1.4).

3.12.2.4 `invoke-ID`

The performer shall return unchanged the invoker-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.12.2.5 `event-identification`

The **`event-identification`** parameter shall identify the event that the user requests be passed to Complex Management by the service provider.

3.12.2.6 `event-list-version`

3.12.2.6.1 The **`event-list-version`** parameter shall specify the version of the event list that the user assumes to be applicable.

3.12.2.6.2 If the provider's Complex Management does not have the same version of the event list, the invocation shall fail with an 'event list version mismatch' `diagnostic`.

3.12.2.7 `event-invocation-identifier`

3.12.2.7.1 The **`event-invocation-identifier`** parameter shall contain a monotonically increasing sequence number and shall be

a) set to '0' for the first FSP-THROW-EVENT invocation following the first successful FSP-BIND to the service instance;

b) incremented for each FSP-THROW-EVENT invocation accepted by the provider.

3.12.2.7.2 The performer shall set the return value of the `event-invocation-identifier` parameter to the sequence number expected in the next FSP-THROW-EVENT operation:

a) if the invocation is accepted, the value of `event-invocation-identifier` in the return shall be one greater than the value in the invoke parameter;

b) if the invocation is rejected, `event-invocation-identifier` in the return shall contain the value expected by the provider.

3.12.2.7.3 The user shall set the value of `event-invocation-identifier` in the next FSP-THROW-EVENT invocation to the value returned by the provider.

3.12.2.7.4 The value expected by the provider shall not be altered when the association between user and provider is reestablished after being released or aborted.

3.12.2.7.5 The provider shall use the user-supplied value of `event-invocation-identifier` to reference the respective event in all subsequent reports.

3.12.2.8 `result`

The **`result`** parameter shall specify the result of the FSP-THROW-EVENT invocation and shall contain one of the following values:

a) 'positive result'—the event has been forwarded to Complex Management and, if applicable, to the SLE Complex providing the underlying FTCF and/or CLTU service;

b) 'negative result'—the provider did not forward the event.

3.12.2.9 `diagnostic`

3.12.2.9.1 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'no such event'—the `event-identification` specified by the user through this FSP-THROW-EVENT operation is not in the set of `event-identification`s recognized by Complex Management;

c) 'event list version mismatch'—the Complex Management event list version is different from the version assumed by the user;

d) 'event condition evaluated to false'—the conditions defined in the event handler did not evaluate to true and therefore no action was taken;

e) 'other'—the reason for rejection of the operation will have to be found by other means.

3.12.2.9.2 If `result` is 'positive result', `diagnostic` shall not be provided.

3.12.3 EFFECTS

3.12.3.1 If the invocation is accepted, the event shall be forwarded to Complex management.

NOTE – The effects of the mission-specific actions performed depend on the event handler and are outside the scope of this specification.

3.12.3.2 The FSP provider shall pass notifications received from the event handler to the user by invoking FSP-ASYNC-NOTIFY.

3.13 FSP-INVOKE-DIRECTIVE

3.13.1 PURPOSE

3.13.1.1 The user shall invoke the FSP-INVOKE-DIRECTIVE operation to invoke the TC Directives, as specified in reference [4], as necessary in order to rapidly (re-)establish the commanding capability.

3.13.1.2 For a given telecommand VC, only one FSP service instance may be configured to allow invocation of the FSP-INVOKE-DIRECTIVE operation.

NOTE – TC Directives cause a re-configuration of the TC protocol entities.

3.13.1.3 The provider shall confirm the FSP-INVOKE-DIRECTIVE invocation to indicate acceptance of the directive for further processing or rejection of the operation.

3.13.1.4 The provider shall use the FSP-ASYNC-NOTIFY operation to relay information to the user concerning completion of the directive.

3.13.1.5 The FSP-INVOKE-DIRECTIVE operation is valid only in state 3 (‘active’) and shall be invoked only by the user.

3.13.2 INVOCATION, RETURN AND PARAMETERS

3.13.2.1 General

The parameters of the FSP-INVOKE-DIRECTIVE operation shall be present in the invocation and return as specified in table 3-15.

Table 3-15: FSP-INVOKE-DIRECTIVE Parameters

Parameters	Invocation	Return
invoker-credentials	M	
performer-credentials		M
invoke-ID	M	M
directive-identification	M	M
directive	M	
result		M
diagnostic		C

3.13.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-INVOKE-DIRECTIVE invocation (see 3.1.4).

3.13.2.3 `performer-credentials`

The **`performer-credentials`** parameter shall provide information that enables the invoker to authenticate the return from the performance of FSP-INVOKE-DIRECTIVE (see 3.1.4).

3.13.2.4 `invoke-ID`

The performer shall return unchanged the invoker-supplied value of the `invoke-ID` parameter (see 3.1.5).

3.13.2.5 `directive-identification`

3.13.2.5.1 The **`directive-identification`** parameter shall contain a monotonically increasing sequence number and shall be

a) set to '0' for the first FSP-INVOKE-DIRECTIVE invocation following the first successful FSP-BIND to the service instance;

b) incremented for each FSP-INVOKE-DIRECTIVE invocation accepted by the provider.

3.13.2.5.2 The performer shall set the return value of the `directive-identification` parameter to the sequence number expected in the next FSP-INVOKE-DIRECTIVE operation:

a) if the invocation is accepted, the value of `directive-identification` in the return shall be one greater than the value in the invoke parameter;

b) if the invocation is rejected, `directive-identification` in the return shall contain the value expected by the provider.

3.13.2.5.3 The user shall set the value of `directive-identification` in the next FSP-INVOKE-DIRECTIVE invocation to the value returned by the provider.

3.13.2.5.4 The value expected by the provider shall not be altered when the association between user and provider is reestablished after being released or aborted.

3.13.2.5.5 The performer shall use the user-supplied value of `directive-identification` to reference the respective event in all subsequent reports.

3.13.2.6 `directive`

3.13.2.6.1 The **`directive`** parameter shall contain the code of the directive and the associated parameter.

NOTE – The detailed type specification is provided in annex A.

3.13.2.6.2 The directives supported are:

a) 'Initiate AD Service (without CLCW check)';

b) 'Initiate AD Service (with CLCW check)';

c) 'Initiate AD Service (with Unlock)';

NOTE – This directive causes a BC segment to be transmitted to the space element.

d) 'Initiate AD Service (with Set V(R))';

NOTE – This directive causes a BC segment to be transmitted to the space element.

e) 'Terminate AD Service';

NOTE – This directive selectively deletes all AD packets currently buffered for transmission on the service instance's VC, but does not delete BD packets (see annex B).

f) 'Resume AD Service';

g) 'Set V(S) to V*(S)';

h) 'Set FOP_Sliding_Window_Width';

i) 'Set T1_Initial';

j) 'Set Transmission_Limit';

k) 'Set Timeout_Type';

l) 'Abort VC';

NOTE – This directive causes the frames that resulted from the segmentation and MAP multiplexing processes and are queued for transmission to the FTCF service to be deleted. This directive will affect all packets and not only AD packets.

m) 'Modify `map-multiplexing-control`'.

NOTE – This directive allows the user to change the priority with which the concurrent MAPs are multiplexed into the service instance's VC. Depending on the applicable multiplexing scheme, either the priority list identifying the priority applicable to each MAP or a polling vector are sent.

3.13.2.7 `result`

The **`result`** parameter shall specify the result of the FSP-INVOKE-DIRECTIVE invocation and shall contain one of the following values:

a) 'positive result'—the provider has accepted the directive for further processing;

b) 'negative result'—the provider is unable to accept the directive for processing.

3.13.2.8 `diagnostic`

3.13.2.8.1 If `result` is 'negative result', `diagnostic` shall be provided and shall contain one of the following values:

a) 'duplicate Invoke-ID'—the value of the `invoke-ID` parameter is the same as the `invoke-ID` value of a previous, outstanding operation;

b) 'directive invocation not allowed'—the operation FSP-INVOKE-DIRECTIVE is not enabled for this service instance;

c) 'unexpected directive identifier'—`directive-identification` is not set to the value expected by the provider;

d) 'directive error'—the provider has found an error in the submitted directive: the directive may be unknown or the parameter associated with the invoked directive is in error (e.g., out of range);

e) 'other'—the reason for rejection of the operation will have to be found by other means.

3.13.2.8.2 If `result` is 'positive result', `diagnostic` shall not be provided.

3.13.3 EFFECTS

If the invocation is accepted, the provider shall forward the TC directive for processing.

3.14 FSP-PEER-ABORT

3.14.1 PURPOSE

3.14.1.1 The user or provider shall invoke the FSP-PEER-ABORT operation to notify the peer system that the local application detected an error that necessitates the association be terminated abnormally.

3.14.1.2 The FSP-PEER-ABORT operation is valid only in states 2 ('ready') and 3 ('active') and may be invoked by either the user or the provider.

NOTE – FSP-PEER-ABORT is an unconfirmed operation.

3.14.2 INVOCATION AND PARAMETERS

3.14.2.1 General

The parameters of the FSP-PEER-ABORT operation shall be present in the invocation and return as specified in table 3-16.

Table 3-16: FSP-PEER-ABORT Parameters

Parameters	Invocation
`invoker-credentials`	M
`diagnostic`	M

3.14.2.2 `invoker-credentials`

The **`invoker-credentials`** parameter shall provide information that enables the performer to authenticate the FSP-PEER-ABORT invocation (see 3.1.4).

3.14.2.3 `diagnostic`

The **`diagnostic`** parameter shall specify why the abort was issued and shall contain one of the following values:

a) 'operational requirement'—the local system had to terminate the communication to accommodate some other operational need;

b) 'protocol error'—the local application detected an error in the sequencing of one or more operations;

c) 'encoding error'—the local application detected an error in the encoding of one or more operation parameters, or did not recognize the operation;

d) 'return timeout'—the local application detected that a return of one operation was not received within a specified time limit;

e) 'end of service provision period'—the local application detected that the service period has ended and the initiator has not invoked an UNBIND operation;

f) 'other'—the local application detected an unspecified error during the processing of one or more operations.

3.14.3 EFFECTS

3.14.3.1 The association shall be aborted and the provider shall return to state 1 ('unbound').

3.14.3.2 All packets buffered for this service instance by the provider shall be discarded.

3.14.3.3 The state of the packet service when an association for this service instance is aborted shall be reset; this also applies to any changes made by means of the FSP-SYNC-SET-PARAMETER operation.

3.14.3.4 Statistical information required for the generation of the status report shall be retained during the service instance provision period.

3.14.3.5 Parameters that serve to relate notifications on an activity to operations that triggered that activity shall not be altered when the association is released.

3.15 FSP-PROTOCOL-ABORT

3.15.1 PURPOSE

3.15.1.1 The underlying communications service shall invoke the FSP-PROTOCOL-ABORT operation to notify the SLE application (user or provider) that communications with the peer system have been disrupted and that the association was terminated abnormally.

NOTE – FSP-PROTOCOL-ABORT is an unconfirmed operation.

3.15.1.2 The FSP-PROTOCOL-ABORT operation is valid only in states 2 ('ready') and 3 ('active') and shall be invoked only by the underlying communications service.

3.15.2 INVOCATION, RETURN AND PARAMETERS

3.15.2.1 General

The parameter of the FSP-PROTOCOL-ABORT operation shall be present in the invocation as specified in table 3-17.

Table 3-17: FSP-PROTOCOL-ABORT Parameter

Parameters	Invocation
`diagnostic`	M

3.15.2.2 `diagnostic`

The **`diagnostic`** parameter shall be used to convey to the SLE provider or user any information provided by the underlying communications service that is descriptive of the specific nature or cause of the fault (e.g., an error code).

NOTE – The value of `diagnostic` is implementation dependent.

3.15.3 EFFECTS

3.15.3.1 The association shall be aborted and the provider shall return to state 1 ('unbound').

3.15.3.2 The provider shall discard all packets buffered for this service instance.

3.15.3.3 The state of the packet service shall be reset; this also applies to any changes made by means of the FSP-SYNC-SET-PARAMETER operation.

3.15.3.4 Statistical information required for the generation of the status report shall be retained during the service instance provision period.

3.15.3.5 Parameters that serve to relate notifications on an activity to operations that triggered that activity shall not be altered when the association is aborted.

4 FSP PROTOCOL

4.1 GENERIC PROTOCOL CHARACTERISTICS

4.1.1 UNEXPECTED PROTOCOL DATA UNIT

In the case that the peer application sends an SLE-PDU not allowed in the current state of the performer, the performer shall process the operation as follows:

a) invocation: abort the association by invoking the FSP-PEER-ABORT operation;

b) return: abort the association by invoking the FSP-PEER-ABORT operation.

4.1.2 UNDECODABLE PROTOCOL DATA UNIT

In case the application receives an invocation or return containing a wrong type and therefore an undecodable SLE-PDU, the association shall be aborted by means of the FSP-PEER-ABORT operation.

4.1.3 UNKNOWN PROTOCOL DATA UNIT

In case the application receives an SLE-PDU of an unknown type, the association shall be aborted by means of invoking the FSP-PEER-ABORT operation.

4.1.4 MISSING RETURN

For confirmed operations:

a) in the event that the invoker does not receive the return from the performer within a period specified by management, the invoker shall abort the association by invoking the FSP-PEER-ABORT operation;

b) the time-out period shall be chosen taking into account performance of user and provider application as well as the delays introduced by the underlying communications service;

c) the generation of the return therefore must not depend on any human interaction.

4.1.5 FAILING AUTHENTICATION

4.1.5.1 An incoming invocation or return shall be ignored under the following conditions:

a) the credentials parameter is ‘unused’ or cannot be authenticated when, by management arrangement, credentials are required;

b) the credentials parameter is not 'unused' when, by management arrangement, credentials are not required.

4.1.5.2 If an invocation is ignored, the operation shall not be performed. If a return is ignored, it shall be as if no report of the outcome of operation has been received.

4.1.6 CONCURRENT PROCESSING

NOTE – The protocol specified in this Recommendation enables the user and provider to interact on the SLE transfer port in a non-blocking way (with the exception of the BIND and UNBIND operations). That is, after having submitted an invocation, the invoker does not need to wait for the associated return to be sent by the performer before the next request can be submitted. The `invoke-ID` value in the return allows the invoker to correlate this return with the corresponding invocation. Since the performer may process the invocations in concurrent threads, the sequence in which the invocations are submitted does not determine the sequence in which the returns will be received.

4.1.6.1 The invoker may opt not to exploit the non-blocking capability and always wait for the return before invoking the next operation, in which case the `invoke-ID` parameter shall take the 'null' value.

4.1.6.2 Compliance with this Recommendation does not require the performer to process requests concurrently; however, the performer must still accept requests from the invoker in non-blocking mode and buffer and serialize them by local means not visible externally.

4.2 STATE TRANSITION TABLE

NOTE – The state table specifies service operation interactions and state transitions for the provider system. The structure of the state table is:

- The leftmost column lists all incoming events. Where these events correspond to the arrival of an incoming SLE-PDU, in addition to a 'verbal' description of the event, the ASN.1 type defined for this SLE-PDU in annex A is indicated in normal parentheses ( ).

- Where an event is internal to the provider, its description is put in quotation marks ' '.

- Where an event is triggered by an unrecognized SLE-PDU, i.e., when the ASN.1 type of the SLE-PDU is unknown ('unknown SLE-PDU') or the SLE-PDU fails to authenticate ('not authenticated PDU'), its description is put in quotation marks.

- The three columns (one column per state) on the right side of the table specify the behavior the provider will assume depending on the current state and the incoming event. In some cases, the behavior in addition depends on Boolean conditions, also referred to as predicates. Such conditions are put in double quotes " ". The dependency is presented in form of an 'IF <condition> THEN <action> [ELSEIF <condition> THEN <action>] ELSE <action>' clause.

- If the action is simply to send a specific SLE-PDU, this is indicated by the ASN.1 type of this SLE-PDU in normal parentheses ( ). The convention has been introduced to use '+' and '-' to represent positive and negative responses respectively. Where several actions are to be taken, this is indicated by a 'compound action' which is put in curly braces { }. The individual actions making up this compound action are identified in the notes below the table.

- 'Not applicable' is stated where the given event can only occur in the given state because of an implementation error on the provider side.

- Where the consequences of an incoming event are not visible to the user because the provider does not send any SLE-PDU in reaction to the given event, the action is put in square brackets [ ].

- State transitions are indicated by an arrow and the number of the state that will be entered; for example, → 1 indicates the transition to state 1. This notation has also been used where actually no state transition takes place.

- The actions to be taken and the state transition are considered to be atomic operations. The sequence is irrelevant except that SLE-PDUs shall be sent in the sequence stated in the table.

Table 4-1: Provider Behavior

Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
BIND invocation (fspBindInvoke)	IF “positive result” THEN → 2 (+fspBindReturn) ELSE → 1 (-fspBindReturn)	{peer abort} → 1	{peer abort} → 1
UNBIND invocation (fspUnbindInvoke)	Not applicable → 1	stop reporting timer → 1 stop all return timers, (fspUnbindReturn)	{peer abort} → 1
START invocation (fspStartInvoke)	Not applicable → 1	IF “positive result” THEN → 3 (+fspStartReturn) ELSE → 2 (-fspStartReturn)	{peer abort} → 1
STOP invocation (fspStopInvoke)	Not applicable → 1	{peer abort} → 1	IF “positive result” THEN → 2 clear packet queues set suspense mode to FALSE (+fspStopReturn) ELSE (-fspStopReturn) → 2
TRANSFER-DATA invocation (fspTransferDataInvoke)	Not applicable → 1	{peer abort} → 1	IF “accept packet” THEN → 3 queue packet, (+fspTransferDataReturn) ELSE → 3 (-fspTransferDataReturn)

Table 4-1: Provider Behavior (Continued)

Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
SCHEDULE-STATUS-REPORT invocation (fspScheduleStatusReportInvoke)	Not applicable → 1	IF “immediately” THEN → 2 {immediate report} ELSE IF “periodically” THEN → 2 (+fspScheduleStatusReportReturn) {periodic report} ELSE IF “stop” THEN → 2 {stop reporting} ELSE → 2 (-fspScheduleStatusReportReturn)	IF “immediately” THEN → 3 {immediate report} ELSE IF “periodically” THEN → 3 (+fspScheduleStatusReportReturn) {periodic report} ELSE IF “stop” THEN → 3 {stop reporting} ELSE → 3 (-fspScheduleStatusReportReturn)
‘reporting timer expired’ event	Not applicable → 1	{periodic report} → 2	{periodic report} → 3
‘return timer n expired’ event	Not applicable → 1	{peer abort} → 1	{peer abort} → 1
SYNC-SET-PARAMETER invocation (fspSyncSetParameterInvoke)	Not applicable → 1	IF “positive result” THEN → 2 queue set operation (+fspSyncSetParameterReturn) ELSE → 2 (-fspSyncSetParameterReturn)	IF “positive result” THEN → 3 queue set operation (+fspSyncSetParameterReturn) ELSE → 3 (-fspSyncSetParameterReturn)
‘parameter n modified’ event	Not applicable → 1	{notify} → 2	{notify} → 3
GET-PARAMETER invocation (fspGetParameterInvoke)	Not applicable → 1	IF “positive result” THEN → 2 (+fspGetParameterReturn) ELSE → 2 (-fspGetParameterReturn)	IF “positive result” THEN → 3 (+fspGetParameterReturn) ELSE → 3 (-fspGetParameterReturn)

Table 4-1: Provider Behavior (Continued)

Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
THROW-EVENT invocation (fspThrowEventInvoke)	Not applicable → 1	IF “positive result” THEN → 2 (+fspThrowEventReturn) ELSE → 2 (-fspThrowEventReturn)	IF “positive result” THEN → 3 (+fspThrowEventReturn) ELSE → 3 (-fspThrowEventReturn)
INVOKE-DIRECTIVE invocation (fspInvokeDirectiveInvoke)	Not applicable → 1	{peer abort} → 1	IF “positive result” THEN → 3 queue directive (+fspInvokeDirectiveReturn) ELSE → 3 (-fspInvokeDirectiveReturn)
‘packet transferred’ event	Not applicable → 1	IF “report transfer” THEN → 2 {notify} ELSE → 2 [ignore]	IF “report transfer” THEN → 3 {notify} ELSE → 3 [ignore]
‘packet radiated’ event	Not applicable → 1	IF “report radiation” THEN → 2 {notify} ELSE → 2 [ignore]	IF “report radiation” THEN → 3 {notify} ELSE → 3 [ignore]
‘sldu expired’ event	Not applicable → 1	Not applicable → 2	{notify and suspend} → 3
‘packet transmission mode mismatch’	Not applicable → 1	Not applicable → 2	{notify and suspend} → 3
‘production interrupted’ event	Not applicable → 1	Not applicable → 2	{notify and suspend} → 3
‘production halted’ event	Not applicable → 1	Not applicable → 2	{notify and suspend} → 3
‘production waiting’ event	Not applicable → 1	{notify} → 2	Not applicable → 3
‘end of data’ event	Not applicable → 1	{notify} → 2	{notify} → 3

Table 4-1: Provider Behavior (Continued)

Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
'no invoke directive capability on this VC' event	Not applicable → 1	{notify} → 2	{notify} → 3
'positive confirm response to directive' event	Not applicable → 1	{notify} → 2	{notify} → 3
'negative confirm response to directive' event	Not applicable → 1	{notify} → 2	{notify} → 3
'suspend' event	Not applicable → 1	{notify} → 2	{notify} → 3
'action list completed' event	Not applicable → 1	{notify} → 2	{notify} → 3
'action list not completed' event	Not applicable → 1	{notify} → 2	{notify} → 3
'event condition evaluated to false' event	Not applicable → 1	{notify} → 2	{notify} → 3
'FOP alert' event	Not applicable → 1	{notify} → 2	{notify} → 3
ASYNC-NOTIFY return	Not applicable → 1	[ignore] → 2 stop return timer <n>	[ignore] → 3 stop return timer <n>
PEER ABORT (fspPeerAbortInvoke)	Not applicable → 1	{clean up} → 1	{clean up} → 1
PROTOCOL ABORT (fspProtocolAbortInvoke)	Not applicable → 1	{clean up} → 1	{clean up} → 1
'unexpected SLE-PDU' event	Not applicable → 1	{peer abort} → 1	{peer abort} → 1
'undecodable SLE-PDU' event	Not applicable → 1	{peer abort} → 1	{peer abort} → 1
'unknown SLE-PDU' event	Not applicable → 1	{peer abort} → 1	{peer abort} → 1
'not authenticated SLE-PDU' event	Not applicable → 1	[ignore] → 2	[ignore] → 3

NOTES:

1 "positive result" evaluates to TRUE if all checks on the invocation are passed.

2 {peer abort} covers the following actions:

a) send (fspPeerAbortInvoke);

b) stop reporting timer;

c) stop all return timers;

d) clear packet queues (only of packets relating to the service instance).

3 'return timer <n> expired' event: this event occurs when the provider has invoked a confirmed operation with `invoke-ID` set to <n> and the return from that operation has not been received within the expected time. Since the interface may be operated in a non-blocking mode, several returns may be pending at any given point in time, even for the same type of operation. Whenever the provider invokes a confirmed operation with `invoke-ID` set to <n>, it shall start an associated return timer <n>. Should this timer expire before the return <n> is received, the provider shall invoke FSP-PEER-ABORT.

4 "accept packet" evaluates to TRUE only if "positive result" evaluates to TRUE and suspense mode is FALSE.

5 "immediately" evaluates to TRUE if all checks on the SCHEDULE-STATUS-REPORT invocation are passed and if the `report-request-type` value is 'immediately'. {immediate report} covers the following actions:

a) send (+fspScheduleStatusReportReturn);

b) send (fspStatusReportInvoke);

c) stop the reporting timer.

6 "periodically" evaluates to TRUE if all checks on the SCHEDULE-STATUS-REPORT invocation are passed and if the `report-request-type` value is 'periodically'. {periodic report} covers the following actions:

a) send (+fspScheduleStatusReportReturn);

b) send (fspStatusReportInvoke);

c) start reporting timer with the time set to the `reporting-cycle` value in the most recent SCHEDULE-STATUS-REPORT invocation.

7 "stop" evaluates to TRUE if all checks on the SCHEDULE-STATUS-REPORT invocation are passed and if the `report-request-type` value is 'stop'. {stop reporting} covers the following actions:

a) send (+fspScheduleStatusReportReturn);

b) stop reporting timer.

8 {notify} covers the following actions:

a) send (fspAsyncNotifyInvoke) with invoke-ID <n>;

b) start return timer <n>.

9 {notify and suspend} covers the following actions:

a) send (fspAsyncNotifyInvoke) with invoke-ID <n>;

b) start return timer <n>;

c) clear packet queues (only of packets relating to the service instance);

d) set suspense mode to TRUE.

10 "report radiation" evaluates to TRUE if the `radiation-report` value was 'produce report' in the FSP-TRANSFER-DATA operation for the packet associated with the radiation event.

11 "report transfer" evaluates to TRUE if the `transfer-report` value was 'produce report' in the FSP-TRANSFER-DATA operation for the packet associated with the transfer event.

12 {clean up} covers the following actions:

a) stop the reporting timer;

b) stop all return timers;

c) clear packet queues (only of packets relating to the service instance).

13 Further descriptions of events can be found in the following subsections:

Event	Reference
‘reporting timer expired’	3.8.2
‘return timer <n> expired’	4.1.4
‘parameter <n> modified’	3.7.2.5
‘packet transferred’	3.7.2.5
‘packet radiated’	3.7.2.5
‘sldu expired’	3.7.2.5
‘packet transmission mode mismatch’	3.7.2.5
‘production interrupted’	3.7.2.5
‘production halted’	3.7.2.5
‘production waiting’	3.7.2.5
‘end of data’	3.7.2.5
‘no invoke directive capability on this VC’	3.7.2.5
‘positive confirm response to directive’	3.7.2.5
‘negative confirm response to directive’	3.7.2.5
‘suspend’	3.7.2.5
‘action list completed’	3.7.2.5
‘action list not completed’	3.7.2.5
‘event condition evaluated to false’	3.7.2.5
‘FOP alert’	3.7.2.5
‘unexpected SLE-PDU’	4.1.1
‘undecodable SLE-PDU’	4.1.2
‘unknown SLE-PDU’ event	4.1.3
‘not authenticated SLE-PDU’ event	4.1.5

ANNEX A

ASN.1 PROTOCOL SPECIFICATION

DATA TYPE DEFINITIONS

(This annex **is** part of the Recommendation)

A1 INTRODUCTION

This annex defines the data types that are used by the FSP service. It is intended to provide a clear specification of these data types and to avoid ambiguity. It is not intended to constrain how these data types are implemented or encoded. These definitions are suitable for inclusion in any type of ASN.1 based protocol that implements the FSP service.

The data type definitions are presented in six ASN.1 modules.

Subsection A2.1 contains basic types which will be common with other SLE transfer services. As further services become specified by CCSDS, further types may be added to this module or existing types may be extended. This does, however, not invalidate the module in its present form because an implementation compliant with a future extended version of this module is still fully interoperable with an implementation based on its present version.

Subsection A2.2 specifies the Protocol Data Units (PDUs) exchanged between an SLE user and an SLE provider application in order to establish, release or abort an association. They are common to all SLE transfer service types.

Subsection A2.3 specifies Protocol Data Units (PDUs) related to operations that are common to SLE transfer service types.

Subsection A2.4 specifies data types that are considered specific for the Forward Space Packet service. In part, these specific types are derived from types specified in Module 1 (A2.1) by means of subtyping.

Subsection A2.5 specifies all incoming (from a provider point of view) PDUs. Where applicable, these PDUs are mapped to the generic PDUs defined in A2.3.

Subsection A2.6 specifies in the same way all outgoing PDUs.

Although A2.2, A2.3, A2.5 and A2.6 define the PDUs that will be exchanged between the SLE provider and user applications, they shall not be understood to require that these PDUs shall completely be mapped to the user data field of the underlying communications protocol. Depending on the communications protocol(s) used, part of the PDUs may be used to determine the appropriate setting of protocol control information.

A 2 FSP DATA TYPE SPECIFICATION

A 2.1 SLE TRANSFER SERVICE COMMON TYPES

```
CCSDS-SLE-TRANSFER-SERVICE-COMMON-TYPES

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

EXPORTS     ConditionalTime
,           Credentials
,           Diagnostics
,           Duration
,           ForwardDuStatus
,           IntPosLong
,           IntPosShort
,           IntUnsignedLong
,           IntUnsignedShort
,           ParameterName
,           ProductionStatus
,           Report
,           InvokeId
,           SpaceLinkDataUnit
,           Time
;
ConditionalTime               ::=   CHOICE
{     undefined               [0]   NULL
,     known                   [1]   Time
}
Credentials                   ::=   CHOICE
{     unused                  [0]   NULL
,     used                    [1]   BIT STRING (SIZE (64 .. 2048))
}

Diagnostics                   ::=   INTEGER
{     duplicateInvokeId       (100)
,     otherReason             (101)
}

-- The Duration is expressed in microseconds
Duration                      ::=   IntUnsignedLong

ForwardDuStatus               ::=   INTEGER
{     radiated                (0)
,     expired                 (1)
,     interrupted             (2)
,     transferred             (3)
-- FSP
,     productionStarted       (4)
,     undefined               (5)
-- FSP
}

-- 1 to (2^32)-1
IntPosLong                    ::=   INTEGER (1 .. 42949667295)

-- 1 to (2^16)-1
IntPosShort                   ::=   INTEGER (1 .. 65535)
```

```
-- 0 to (2^32)-1
IntUnsignedLong               ::=   INTEGER (0 .. 42949667295)

-- 0 to (2^16)-1
IntUnsignedShort              ::=   INTEGER (0 .. 65535)

InvokeId                      ::= IntUnsignedShort

ParameterName                 ::=       INTEGER
{       aggregation-usage           (0)
,       buffer-size                 (1)
,       cop-in-effect               (2)
,       directive-identification    (3)
,       fecf-usage                  (4)
,       fop-sliding-window          (5)
,       fop-state                   (6)
,       frame-Quality               (7)
,       latency-Limit               (8)
,       map-mux-scheme              (9)
,       map-mux-control             (10)
,       maximum-packet-size         (11)
,       maximum-segment-size        (12)
,       maximum-frame-size          (13)
,       segmentation                (14)
,       sent-queue-size             (15)
,       transmission-limit          (16)
,       timer-initial               (17)
,       timeout-type                (18)
,       vc-mux-scheme               (19)
,       vc-mux-control              (20)
,       virtual-channel             (21)
}

ProductionStatus              ::=   INTEGER
{     halted                  (0)
,     waiting                 (1)
,     running                 (2)
}

Report                        ::=   INTEGER
{     produceReport           (0)
,     doNotProduceReport      (1)
}

SpaceLinkDataUnit             ::=   OCTET STRING (SIZE (7 .. 65536))

Time                          ::=   CHOICE
{     ccsdsFormat             [0]   TimeCCSDS
}

TimeCCSDS                     ::=   OCTET STRING (SIZE(8))
-- P-field is implicit (not present, defaulted to 41 hex
-- T-field:
-- 2 octets: number of days since 1958/01/01 00:00:00
-- 4 octets: number of milliseconds of the day
-- 2 octets: number of microseconds of the millisecond
--           (set to 0 if not used)
-- This definition reflects exactly the format of the CCSDS defined
-- time tag as used in spacelink data units.

END
```

A 2.2 SLE TRANSFER SERVICE BIND TYPES

```
CCSDS-SLE-TRANSFER-SERVICE-BIND-TYPES

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

EXPORTS     Sle-Bind-Invoke
,           Sle-Bind-Return
,           Sle-Peer-Abort
,           Sle-Protocol-Abort
,           Sle-Unbind-Invoke
,           Sle-Unbind-Return
;

IMPORTS     Credentials
,           IntPosShort
,           IntUnsignedShort
      FROM  CCSDS-SLE-TRANSFER-SERVICE-COMMON-TYPES
;

-- ============================================================
-- The first part of the module definition contains the SLE-PDUs
-- ============================================================

Sle-Bind-Invoke               ::=   SEQUENCE
{     invokerCredentials            Credentials
,     initiatorPortIdentifier       PortId
,     responderPortIdentifier       PortId
,     serviceType                   ServiceType
,     versionNumber                 VersionNumber
,     serviceInstanceIdentifier     ServiceInstanceIdentifier
,     applicationContext            ApplicationContext
}

Sle-Bind-Return               ::= SEQUENCE
{     performerCredentials    Credentials
,     result                  CHOICE
      {     positive          [2]   SEQUENCE
            {     versionNumber           VersionNumber
            ,     applicationContext      ApplicationContext
            }
      ,     negative          [3]   BindDiagnostic
      }
}

Sle-Peer-Abort                ::=   SEQUENCE
{     invokerCredentials      Credentials
,     diagnostic              PeerAbortDiagnostic
}

Sle-Protocol-Abort            ::= ProtocolAbortDiagnostic

Sle-Unbind-Invoke             ::=   SEQUENCE
{     invokerCredentials      Credentials
,     unbindReason            UnbindReason
```

```
}

Sle-Unbind-Return            ::= Credentials

-- ============================================================
-- The second part of the module definition contains the types
-- used by the SLE-PDUs declared in the first part.
-- ============================================================

ApplicationContext      ::=   CHOICE
{     osiApplicationContext         [0] Context
,     dceApplicationContext         [1] OCTET STRING (SIZE (0 .. 128))
,     genericApplicationContext     [2] OCTET STRING (SIZE (1 .. 128))
,     fixedApplicationContext       [3] NULL
}

ApplicationIdentifier         ::=   INTEGER
{     rtnAllFrames            (0)
,     rtnInsert               (1)
,     rtnMcFrames             (2)
,     rtnVcFrames             (3)
,     rtnVcFsh                (4)
,     rtnVcOcf                (5)
,     rtnMcFsh                (6)
,     rtnMcOcf                (7)
,     rtnBitstr               (8)
,     rtnAosSpacePkt          (9)
,     fwdSpacePkt             (10)
,     fwdVca                  (11)
,     fwdBitstr               (12)
,     fwdProtoVcdu            (13)
,     fwdInsert               (14)
,     fwdCVcdu                (15)
,     fwdTcSpacePkt           (16)
,     fwdTcVca                (17)
,     fwdTcFrame              (18)
,     fwdCltu                 (19)
}

-- This type uses the ANY DEFINED BY feature.
-- That feature is not supported in the 94 ASN.1 !!
AttributeValueAssertion       ::=   SEQUENCE
{     attributeId                   OBJECT IDENTIFIER
,     attributeValue                ANY DEFINED BY attributeId
}

BindDiagnostic                ::=   INTEGER
{     noSuchServiceInstance               (0)
,     invalidTime                         (1)
,     unableToComply                      (2)
,     inconsistentServiceType             (3)
,     versionNotSupported                 (4)
,     applicationContextNotSupported      (5)
,     otherReason                         (1000)
}

Context                       ::=   SEQUENCE
{     serviceSyntax                       Syntax
,     supplementaryInformationSyntax      Syntax
```

```
,     telecommunicationServices           TelecomService
}

DcePort                          ::=   SEQUENCE
{     ipAddress   CHOICE
            {ipv4 [0]   OCTET STRING (SIZE (4))
            ,ipv6 [1]   OCTET STRING (SIZE (16))
            }
,     endPoint    OCTET STRING (SIZE (1 .. 128))
}

GenericPort                      ::=   OCTET STRING (SIZE (1..128))

OsiAddress                       ::=   SEQUENCE
{     presentationSelector    OCTET STRING (SIZE (0 .. 4))
,     sessionSelector         OCTET STRING (SIZE (0 .. 16))
,     transportSelector       OCTET STRING (SIZE (0 .. 16))
,     networkSelector         OCTET STRING (SIZE (0 .. 16))
}

PeerAbortDiagnostic              ::=   INTEGER
{     operationalRequirement              (0)
,     protocolError                       (1)
,     pduEncodingError                    (2)
,     returnTimeout                       (3)
,     endOfServiceProvisionPeriod         (4)
,     otherReason                         (1000)
}

PortId                           ::=   CHOICE
{     osiInitiatorPort        [0]     OsiAddress
,     dceInitiatorPort        [1]     DcePort
,     genericInitiatorPort    [2]     GenericPort
}
-- A given implementation only needs to implement the
-- choice(s) corresponding to the underlying communications
-- technologies actually used by this implementation.

ProtocolAbortDiagnostic          ::=   CHOICE
{     OCTET STRING (SIZE (1 .. 128))
}

RDNSequence                      ::=   SEQUENCE OF
                         RelativeDistinguishedName     (SIZE (5 .. 256))

RelativeDistinguishedName        ::=   SET SIZE (1 .. MAX) OF
                                       AttributeValueAssertion

SessionRequirements              ::=   OCTET STRING (SIZE ( 0 .. 256 ))

ServiceInstanceIdentifier        ::= RDNSequence

ServiceType                      ::=   CHOICE
{     osiApplication          [0]   OBJECT IDENTIFIER
,     ipApplication           [1]   ApplicationIdentifier
```

```
,     genericApplication      [2]   OCTET STRING (SIZE (1 .. 128))
}
--    A given implementation only needs to implement the
--    choice(s) corresponding to the underlying communications
--    technologies actually used by this implementation.

Syntax                        ::=   SEQUENCE
{     syntaxID                IntUnsignedShort
,     abstractSyntax          OBJECT IDENTIFIER
,     transferSyntax          OBJECT IDENTIFIER
}

TelecomService                ::=   SEQUENCE
{     sessionRequ             SessionRequirements
}

UnbindReason                  ::=   INTEGER
{     end                     (0)
,     suspend                 (1)
,     versionNotSupported     (2)
,     other                   (1000)
}

VersionNumber                 ::=   IntPosShort

END
```

A 2.3 SLE TRANSFER SERVICE COMMON PDUS

```
CCSDS-SLE-TRANSFER-COMMON-PDUS

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

EXPORTS     Sle-Acknowledgement
,           Sle-Schedule-Status-Report-Invoke
,           Sle-Schedule-Status-Report-Return
,           Sle-Stop-Invoke
;

IMPORTS     Credentials
,           Diagnostics
,           IntUnsignedShort
,           InvokeId
      FROM  CCSDS-SLE-TRANSFER-SERVICE-COMMON-TYPES
;

-- ============================================================
-- The first part of the module definition contains the SLE-PDUs
-- ============================================================
Sle-Acknowledgement           ::=   SEQUENCE
{     credentials             Credentials
,     invokeId                InvokeId
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   Diagnostics
      }
}

Sle-Schedule-Status-Report-Invoke   ::=   SEQUENCE
{     userCredentials         Credentials
,     invokeId                InvokeId
,     reportRequestType       ReportRequestType
}

Sle-Schedule-Status-Report-Return   ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticScheduleStatusReport
      }
}

Sle-Stop-Invoke               ::=   SEQUENCE
{     userCredential          Credentials
,     invokeId                InvokeId
}
```

```
-- ============================================================
-- The second part of the module definition contains the types
-- used by the SLE-PDUs declared in the first part.
-- ============================================================

DiagnosticScheduleStatusReport      ::=   CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     alreadyStopped          (0)
      ,     invalidReportingCycle   (1)
      }
}

-- The cycle duration is expressed in seconds
ReportingCycle                ::=   INTEGER (60 .. 600)

ReportRequestType             ::=   CHOICE
{     immediately             [0]   NULL
,     periodically            [1]   ReportingCycle
,     stop                    [2]   NULL
}

END
```

A 2.4 FSP TRANSFER STRUCTURES

```
CCSDS-SLE-TRANSFER-FSP-STRUCTURES

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

EXPORTS     AggregationUsage
,           BufferSize
,           FspData
,           FspPacketHeader
,           FspGetParameter
,           FspNotification
,           FspPacketStatus
,           FspParameterName
,           FspSetParameter
,           DiagnosticFspGet
,           DiagnosticFspInvokeDirective
,           DiagnosticFspSet
,           DiagnosticFspStart
,           DiagnosticFspThrowEvent
,           DiagnosticFspTransferData
,           Map
,           MapMuxControl
,           ProductionTime
,           TransmissionMode
;

IMPORTS     Diagnostics
,           ForwardDuStatus
,           IntPosLong
,           IntPosShort
,           IntUnsignedLong
,           InvokeId
,           ParameterName
,           SpaceLinkDataUnit
,           Time
      FROM CCSDS-SLE-TRANSFER-SERVICE-COMMON-TYPES
;

AbsolutePriority              ::=  SEQUENCE
{     mapId                   INTEGER (0 ..63)
,     priority                Priority
}

AggregationUsage              ::=   INTEGER
{     permitted               (0)
,     notPermitted            (1)
}

-- The minimum Telecommand size is 7 octets
-- The maximum Telecommand size is 64 K-Octets + overhead.
BufferSize                    ::=   INTEGER (7 .. 66560)
```

```
DiagnosticFspGet              ::=   CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     unknownParameter        (0)
      }
}

DiagnosticFspInvokeDirective  ::=   CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     directiveInvocationNotAllowed (0)
      ,     unexpectedDirectiveIdentifier (1)
      ,     directiveError                (2)
      }
}

DiagnosticFspSet              ::=   CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     unknownParameter        (0)
      ,     invalidParameter        (1)
      ,     valueOutOfRange         (2)
      }
}

DiagnosticFspStart            ::=   CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     outOfService            (0)
      ,     unableToComply          (1)
      ,     productionTimeExpired   (2)
      }
}

DiagnosticFspThrowEvent       ::=  CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     noSuchEvent                   (0)
      ,     eventConditionEvaluatedFalse  (1)
      ,     eventListVersionMismatch      (2)
      }
}

DiagnosticFspTransferData     ::=   CHOICE
{     common                  [0]   Diagnostics
,     specific                [1]   INTEGER
      {     unableToProcess                     (0)
      ,     invalidTransmissionMode             (1)
      ,     inconsistentProductionTimeRange     (2)
      ,     invalidProductionTime               (3)
      ,     lateSldu                            (4)
      ,     packetTooLong                       (5)
      ,     unableToStore                       (6)
      ,     unsupportedPacketVersion            (7)
      ,     incorrectPacketType                 (8)
      ,     invalidPacketApid                   (9)
      ,     invalidMAP                          (10)
      }
}
```

```
FopAlert                        ::=   INTEGER
{     limit                     (0)
,     lockOutDetected           (1)
,     synch                     (2)
,     invalidNR                 (3)
,     clcw                      (4)
,     lowerLayerOutOfSync       (5)
,     terminateADService        (6)
}

FspData                         ::= SpaceLinkDataUnit

FspGetParameter                 ::=   CHOICE
{     parVcMuxScheme            [0]   SEQUENCE
      {     parameterName             ParameterName (vc-mux-scheme)
      ,     parameterValue            MuxScheme
      }
,     parVcMuxControl           [1]   SEQUENCE
      {     parameterName             ParameterName (vc-mux-control)
      ,     parameterValue            MuxControl
      }
,     parMapMuxScheme           [2]   SEQUENCE
      {     parameterName             ParameterName (map-mux-scheme)
      ,     parameterValue            MuxScheme
      }
,     parMapMuxControl          [3]   SEQUENCE
      {     parameterName             ParameterName (map-mux-control)
      ,     parameterValue            MuxControl
      }
,     directiveIdentification [4]   SEQUENCE
{     parameterName      ParameterName (directive-identification)
      ,     parameterValue            InvokeId
      }
,     parCopInEffect            [5]   SEQUENCE
      {     parameterName             ParameterName (cop-in-effect)
      ,     parameterValue            INTEGER
                                      {     undefined   (0)
                                      ,     cop1        (1)
                                      }
      }
,     parVirtualChannel         [6]   SEQUENCE
      {     parameterName             ParameterName (virtual-channel)
      ,     parameterValue            INTEGER (0 .. 63)
      }
,     parSegmentation           [7]   SEQUENCE
      {     parameterName             ParameterName (segmentation)
      ,     parameterValue            INTEGER
                                      {     segment-header-present  (0)
                                      ,     no-segmentation-applied (1)
                                      }
      }
,     parMaximumPacketSize      [8]   SEQUENCE
      {     parameterName             ParameterName (maximum-packet-size)
      ,     parameterValue            INTEGER (7 .. 65542)
      }
```

```
,     parFecfUsage            [9]   SEQUENCE
      {     parameterName           ParameterName (fecf-usage)
      ,     parameterValue          INTEGER
                                    {     used        (0)
                                    ,     notUsed     (1)
                                    }
      }
,     parAggregationUsage     [10]  SEQUENCE
      {     parameterName           ParameterName (aggregation-usage)
      ,     aggregationUsage        AggregationUsage
      }
,     parMaximumSegmentSize   [11]  SEQUENCE
      {     parameterName          ParameterName (maximum-segment-size)
      ,     parameterValue          INTEGER (7 .. 1019) -- octets
      }
,     parMaximumFrameSize     [12]  SEQUENCE
      {     parameterName           ParameterName (maximum-frame-size)
      ,     parameterValue          INTEGER (12 .. 1026) -- octets
      }
,     parTransmissionLimit    [13]  SEQUENCE
      {     parameterName           ParameterName (transmission-limit)
      ,     parameterValue          INTEGER (1 .. 255)
      }
,     parTimerInitial         [14]  SEQUENCE
      {     parameterName           ParameterName (timer-initial)
      ,     parameterValue          IntPosLong
      }
,     parTimeoutType          [15]  SEQUENCE
      {     parameterName           ParameterName (timeout-type)
      ,     parameterValue          INTEGER
                                    {     generateAlert             (0)
                                    ,     suspendSequenceControlled (1)
                                    }
      }
,     parSentQueueSize        [16]  SEQUENCE
      {     parameterName           ParameterName (sent-queue-size)
      ,     parameterValue          IntPosLong
      }
,     parFopWindow            [17]  SEQUENCE
      {     parameterName           ParameterName (fop-sliding-window)
      ,     parameterValue          INTEGER (1 .. 255)
      }
,     parFopState             [18]  SEQUENCE
      {     parameterName           ParameterName (fop-state)
      ,     fopState                INTEGER
                                    {     active                  (0)
                                    ,     retransmitWithoutWait   (1)
                                    ,     retransmitWithWait      (2)
                                    ,     initialisingWithoutBCFrame (3)
                                    ,     initialisingWithBCFrame (4)
                                    ,     initial                 (5)
                                    }
      }
}
```

```
FspNotification               ::=   CHOICE
{     packetRadiated                        [0]   NULL
,     packetTransferred                     [1]   NULL
,     slduExpired                           [2]   NULL
,     packetTransmissionModeMismatch        [3]   NULL
,     productionInterrupted                 [4]   NULL
,     productionHalted                      [5]   NULL
,     productionWaiting                     [6]   NULL
,     noInvokeDirectiveCapabilityOnthisVC   [7]   NULL
,     endOfData                             [8]   NULL
,     positiveConfirmResponceToDirective    [9]   NULL
,     negativeConfirmResponseToDirective    [10]  NULL
,     parameterModified                     [11]  NULL
,     suspend                               [12]  NULL
,     actionListCompleted                   [13]  NULL
,     actionListNotCompleted                [14]  NULL
,     eventConditionEvaluatedToFalse        [15]  NULL
,     fopAlert                              [16]  FopAlert
}

FspPacketHeader               ::=   CHOICE
{     noPacketProcessed       [0]   NULL
,     packetIdentification    [1]   OCTET STRING (SIZE (6))
}

FspPacketStatus               ::= ForwardDuStatus
(     transferred
|     radiated
|     productionStarted
|     expired
|     interrupted
|     undefined
)

FspParameterName              ::= ParameterName
(     vc-mux-scheme
|     vc-mux-control
|     map-mux-scheme
|     map-mux-control
|     directive-identification
|     cop-in-effect
|     virtual-channel
|     segmentation
|     maximum-packet-size
|     fecf-usage
|     aggregation-usage
|     maximum-segment-size
|     maximum-frame-size
|     transmission-limit
|     timer-initial
|     timeout-type
|     sent-queue-size
|     fop-sliding-window
|     fop-state
)

FspSetParameter               ::=   CHOICE
{     directiveIdentification [1]   SEQUENCE
      {     parameterName     ParameterName (directive-identification)
      ,     parameterValue    InvokeId
```

```
      }
}

Map                           ::=   CHOICE
{     noSegmentHeaderGeneration     [0]   NULL
,     segmentHeaderGeneration       [1]   SEQUENCE
      {     mapIdentifier     INTEGER (0 .. 63)
      ,     resetMap          INTEGER
            {     yes         (0)
            ,     no          (1)
            }
      }
}

MapId                         ::= INTEGER (0 .. 63)

MapMuxControl                 ::= CHOICE
{     absolutePriority  [0]   SEQUENCE SIZE (1 .. 64) OF AbsolutePriority
,     pollingVector     [1]   SEQUENCE SIZE (1 .. 192) OF MapId
}

MuxControl                    ::=   CHOICE
{     muxSchemeIsFifo         [0]   NULL
,     muxSchemeIsPriority     [1]   SEQUENCE SIZE (64) OF Priority
,     muxSchemeIsVector       [2]   SEQUENCE SIZE (192) OF Vector
}

MuxScheme                     ::=   INTEGER
{     fifo                    (0)
,     absolute-priority       (1)
,     polling-vector          (2)
}

-- Highest priority: 1
-- Lowest priority: 64
Priority                      ::= INTEGER (1 .. 64)

ProductionTime                ::=   CHOICE
{     unspecified             [0]   NULL
,     specified               [1]   Time
}

TransmissionMode              ::=   INTEGER
{     sequenceControlled      (0)         -- AD mode
,     expedited               (1)         -- BD mode
}

Vector                        ::=   INTEGER (0 .. 63)

END
```

A 2.5 FSP INCOMING PDUS

```
CCSDS-SLE-TRANSFER-FSP-INCOMING-PDUS

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

IMPORTS     Credentials
,           Duration
,           Report
,           IntPosLong
,           IntPosShort
,           InvokeId
      FROM CCSDS-SLE-TRANSFER-SERVICE-COMMON-TYPES

            Sle-Acknowledgement
,           Sle-Schedule-Status-Report-Invoke
,           Sle-Stop-Invoke
      FROM CCSDS-SLE-TRANSFER-COMMON-PDUS

            AggregationUsage
,           FspData
,           FspParameterName
,           FspSetParameter
,           Map
,           MapMuxControl
,           ProductionTime
,           TransmissionMode
      FROM CCSDS-SLE-TRANSFER-FSP-STRUCTURES

            Sle-Bind-Invoke
,           Sle-Peer-Abort
,           Sle-Protocol-Abort
,           Sle-Unbind-Invoke
      FROM  CCSDS-SLE-TRANSFER-SERVICE-BIND-TYPES
;

-- ============================================================
-- The first part of the module definition contains the FSP type
-- that contains all the possible PDUs the provider may receive.
-- ============================================================

Fsp-User-to-Provider-Pdu            ::=   CHOICE
{     fsp-Bind-Invoke                     [100] Sle-Bind-Invoke

,     fsp-Unbind-Invoke                   [102] Sle-Unbind-Invoke

,     fsp-Start-Invoke                    [0]   FspStartInvoke

,     fsp-Stop-Invoke                     [1]   Sle-Stop-Invoke

,     fsp-Transfer-Data-Invoke            [2]   FspTransferDataInvoke

,     fsp-Async-Notify-Return             [3]   Sle-Acknowledgement

,     fsp-Schedule-Status-Report-Invoke   [4]
```

```
                                    Sle-Schedule-Status-Report-Invoke

,     fsp-Sync-Set-Parameter-Invoke       [5]FspSyncSetParameterInvoke

,     fsp-Get-Parameter-Invoke            [6]   FspGetParameterInvoke

,     fsp-Throw-Event-Invoke              [7]   FspThrowEventInvoke

,     fsp-Invoke-Directive-Invoke         [8] FspInvokeDirectiveInvoke

,     fsp-Peer-Abort-Invoke               [104] Sle-Peer-Abort

-- The protocol abort PDU can only be issued by the
-- communication and is received by the SLE provider.
,     protocol-Abort-Invoke               [105] Sle-Protocol-Abort
}

-- ============================================================
-- The second part of the module definition contains the types
-- used by the FSP-PDUs declared in the first part.
-- ============================================================

FspGetParameterInvoke         ::=   SEQUENCE
{     invokerCredentials      Credentials
,     invokeId                InvokeId
,     fspParameterName        FspParameterName
}

FspInvokeDirectiveInvoke      ::=   SEQUENCE
{     invokerCredentials      Credentials
,     invokeId                InvokeId
,     directiveIdentification InvokeId
,     directive               CHOICE
      {     initiateADserviceWithoutCLCW  [0]   NULL
      ,     initiateADserviceWithCLCW     [1]   NULL
      ,     initiateADserviceWithUnlock   [2]   NULL
      ,     initiateADserviceWithSetVR    [3]   INTEGER (0 .. 255)
      ,     terminateADservice            [4]   NULL
      ,     resumeADservice               [5]   NULL
      ,     setVS                         [6]   INTEGER (0 .. 255)
          ,     setFOPslidingWindowWidth      [7]   INTEGER (1 .. 255)
      ,     setT1Initial                  [8]   IntPosLong
      ,     setTransmissionLimit          [9]   IntPosShort
      ,     setTimeoutType                [10]  INTEGER
                                          {     terminateAD (0)
                                          ,     suspendAD   (1)
                                          }
      ,     abortVC                       [11]  NULL
      ,     modifyMapMuxControl           [12]  MapMuxControl
      }
}

FspStartInvoke                ::=   SEQUENCE
{     invokerCredentials      Credentials
,     invokeId                InvokeId
}

FspSyncSetParameterInvoke     ::=   SEQUENCE
{     invokerCredentials      Credentials
```

```
,     invokeId                InvokeId
,     fspParameter            FspSetParameter
,     setIdentifier           InvokeId
}

FspThrowEventInvoke           ::=   SEQUENCE
{     invokerCredentials      Credentials
,     invokeId                InvokeId
,     eventIdentification     IntPosShort
,     eventListVersion        IntPosShort
,     eventInvokeId           InvokeId
}

FspTransferDataInvoke         ::=   SEQUENCE
{     invokerCredentials      Credentials
,     invokeId                InvokeId
,     earliestProductionTime  ProductionTime
,     latestProductionTime    ProductionTime
,     delayTime               Duration
,     transmissionMode        TransmissionMode
,     map                     Map
,     aggregation             AggregationUsage
,     radiationReport         Report
,     transferReport          Report
,     fspData                 FspData
}

END
```

A 2.6 FSP OUTGOING PDUS

```
CCSDS-SLE-TRANSFER-FSP-OUTGOING-PDUS

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

IMPORTS     Credentials
,           IntUnsignedLong
,           InvokeId
,           ProductionStatus
,           Time
      FROM CCSDS-SLE-TRANSFER-SERVICE-COMMON-TYPES

            Sle-Acknowledgement
,           Sle-Schedule-Status-Report-Return
      FROM CCSDS-SLE-TRANSFER-COMMON-PDUS

            BufferSize
,           DiagnosticFspGet
,           DiagnosticFspInvokeDirective
,           DiagnosticFspSet
,           DiagnosticFspStart
,           DiagnosticFspThrowEvent
,           DiagnosticFspTransferData
,           FspPacketHeader
,           FspGetParameter
,           FspNotification
,           FspPacketStatus
,           ProductionTime
      FROM CCSDS-SLE-TRANSFER-FSP-STRUCTURES

            Sle-Bind-Return
,           Sle-Peer-Abort
,           Sle-Protocol-Abort
,           Sle-Unbind-Return
      FROM CCSDS-SLE-TRANSFER-SERVICE-BIND-TYPES
;

-- ============================================================
-- The first part of the module definition contains the FSP type
-- that contains all the possible PDUs the provider may send.
-- ============================================================

Fsp-Provider-to-User-Pdu      ::=   CHOICE
{     fsp-Bind-Return                     [101] Sle-Bind-Return

,     fsp-Unbind-Return                   [103] Sle-Unbind-Return

,     fsp-Start-Return                    [0]   FspStartReturn

,     fsp-Stop-Return                     [1]   Sle-Acknowledgement

,     fsp-Transfer-Data-Return            [2]   FspTransferDataReturn

,     fsp-Async-Notify-Invoke             [3]   FspAsyncNotifyInvoke
```

```
,     fsp-Schedule-Status-Report-Return   [4]
                                    Sle-Schedule-Status-Report-Return

,     fsp-Status-Report-Invoke            [5]   FspStatusReportInvoke

,     fsp-Sync-Set-Parameter-Return       [6]FspSyncSetParameterReturn

,     fsp-Get-Parameter-Return            [7]   FspGetParameterReturn

,     fsp-Throw-Event-Return              [8]   FspThrowEventReturn

,     fsp-Invoke-Directive-Return         [9] FspInvokeDirectiveReturn

,     fsp-Peer-Abort-Invoke               [104] Sle-Peer-Abort

-- The protocol abort PDU can only be issued by the
-- communication and is received by the SLE user.
,     protocol-Abort-Invoke               [105] Sle-Protocol-Abort

}

-- ============================================================
-- The second part of the module definition contains the types
-- used by the FSP-PDUs declared in the first part.
-- ============================================================

FspAsyncNotifyInvoke    ::=   SEQUENCE
{     invokerCredentials      Credentials
                ,     invokeId                InvokeId
,     fspNotification         FspNotification
,     setExecutedId           InvokeId
,     directiveExecutedId     InvokeId
,     eventThrownId           InvokeId
,     fspPacketHeaderLastOk   FspPacketHeader
,     fspPacketHeaderLastProc FspPacketHeader
,     productionStartTime     Time
,     productionStopTime      Time
,     fspPacketStatus         FspPacketStatus
,     productionStatus        ProductionStatus
}

FspInvokeDirectiveReturn      ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     directiveIdentification InvokeId
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticFspInvokeDirective
      }
}

FspGetParameterReturn         ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     fspGetParameter         FspGetParameter
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticFspGet
```

```
      }
}

FspStartReturn                ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     startProductionTime     Time
,     stopProductionTime      Time
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticFspStart
      }
}

FspStatusReportInvoke         ::=   SEQUENCE
{     invokerCredentials            Credentials
,     fspPacketHeaderLastOk         FspPacketHeader
,     fspPacketHeaderLastProc       FspPacketHeader
,     processingStartTime           ProductionTime
,     processingStopTime            ProductionTime
,     reportPacketStatus            FspPacketStatus
,     productionStatus              ProductionStatus
,     numberOfPacketsReceived       IntUnsignedLong
,     numberOfPacketsProcessed      IntUnsignedLong
,     numberOfPacketsRadiated       IntUnsignedLong
,     numberOfPacketsTransferred    IntUnsignedLong
,     fspBufferAvailable            BufferSize
}

FspSyncSetParameterReturn     ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     setIdentifier           InvokeId
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticFspSet
      }
}

FspThrowEventReturn           ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     eventInvokeId           InvokeId
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticFspThrowEvent
      }
}

FspTransferDataReturn         ::=   SEQUENCE
{     performerCredentials    Credentials
,     invokeId                InvokeId
,     packetBufferAvailable   BufferSize
,     result                  CHOICE
      {     positiveResult    [0]   NULL
      ,     negativeResult    [1]   DiagnosticFspTransferData
      }
}

END
```

ANNEX B

FSP QUEUES AND MULTIPLEXING

BEHAVIOR DEFINITION

(This annex **is** part of the Recommendation)

B 1 INTRODUCTION

This annex provides a logical view of the TC System. Guidelines are established to ensure that the behavior of an FSP Service provider is independent of the actual implementation.

The FSP provider serves several Users concurrently, coordinates and multiplexes their packets. Figure B-1 presents a typical scenario for the FSP Service. Service Provision is carried out on many Service Instances which share one production engine. It is worth remarking that such a production engine is also possibly shared with TC VCA Service Instances but on a given TC Virtual Channel, only one type of Service Instance is allowed.

NOTE – The following descriptions and figures represent a logical view of the TC System, and physical implementation may not necessarily correspond to the data structures implied by the following descriptions and figures.

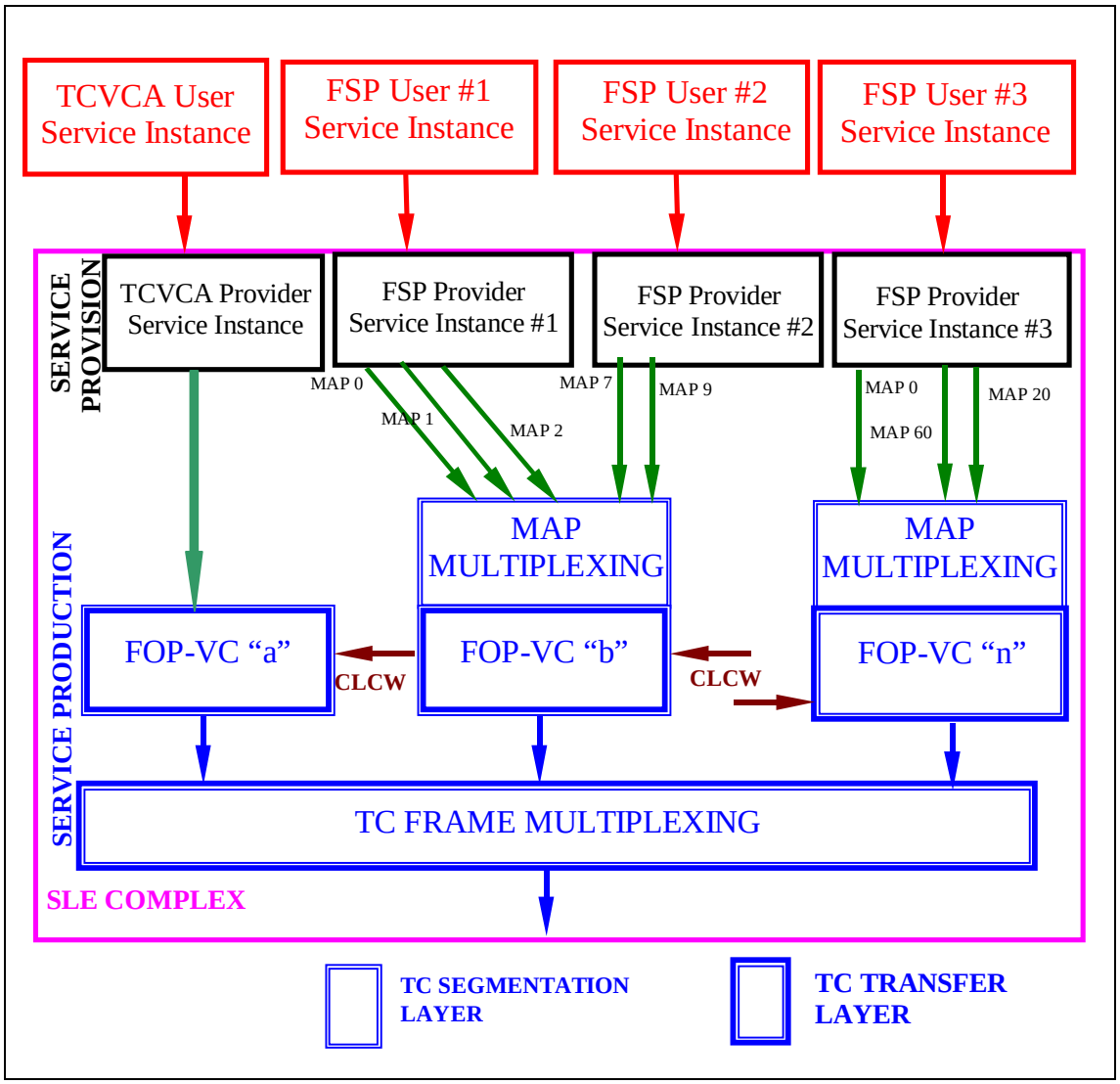


Figure B-1: FSP: MAP and TC Frame Multiplexing

B 2 MULTIPLEXING WITHIN THE FSP SERVICE

Two levels of Multiplexing are provided within the FSP Service, i.e.,

– multiplexing of TC Frame Data Units [1] [Reference TC Part 2] (MAP Multiplexing);

[1] Five different possible TC Frame Data Units types exist: TC Segments containing a portion of one TC User Data Unit; TC Segments containing one complete TC User Data Unit; TC Segments containing multiple complete TC Packets; TC Frame Data Units containing one complete TC User Data Unit; and TC Frame Data Units containing multiple complete TC Packets. The last two types do not really allow multiplexing since the segment header, including the MAP Identifier, is missing. The same is true also for the first three cases whenever MAPs are not used and the MAP

– multiplexing of TC Frames.

TC Frame Multiplexing is carried out by the Telecommand Transfer Layer and is therefore applicable to the TC-VCA Service as well. In turn, MAP Multiplexing is carried out by the Telecommand Segmentation Layer and is therefore applicable only to the FSP Service. The following types of Multiplexing Schemes, applicable to MAP Multiplexing as well as to TC Frame Multiplexing, are defined:

– First In First Out (FIFO);

– Absolute Priority;

– Polling Vector.

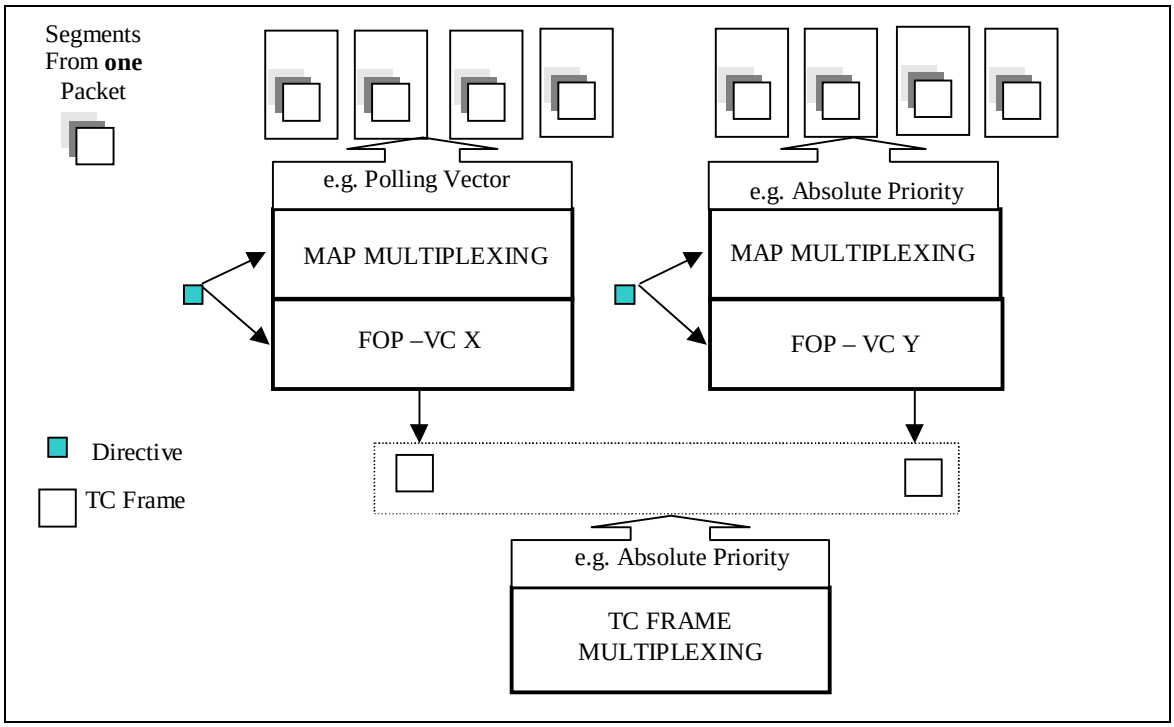


Figure B-2: MAP and VC Multiplexing

First In First Out (FIFO): The requests are served in the same order they are received from the User.

Absolute Priority: The requests are served always selecting that one with highest priority (all priorities shall be different).

Identifier can only assume a single fixed value. In all the cases for which multiplexing is either impossible or not applicable, the only allowed Multiplexing Scheme is the FIFO type.

Polling Vector: The requests are served according to a predefined polling vector.

All the Schemes are independent of each other and, at a given time, different schemes for the actual multiplexing can be in use as shown in figure B-2.

B 3 QUEUE HANDLING IN THE FSP SERVICE

A number of 'logical' queues are needed to handle FSP Requests, Packets and Segments. The number depends on the MAP Multiplexing scheme applied to a given VC as described in the following subsections.

Directives are inserted in a Directives Queue for immediate execution. Only the Service Instance in charge of FOP Management is allowed to issue Directives.

B 3.1 Queues for FIFO Multiplexing

The MAP Multiplexing scheme of the 'FIFO' type is described in figure B-3. As an example, two Service Instances (operating on the same VC) are shown. Each Service Instance has its own Requests Queue. Packets are taken out from the Requests Queue only when the Earliest Production Time is reached. Only when the first packet in a queue is removed is the next packet examined for its Earliest Production Time. Therefore, subsequent Packets wait in the queue until the first one is removed. Once removed from the front of the Requests Queue, the packet passes into the back of a Packets Queue. The first packet in the Packets Queue is segmented (if needed) and the segments pass into a Segments Queue.

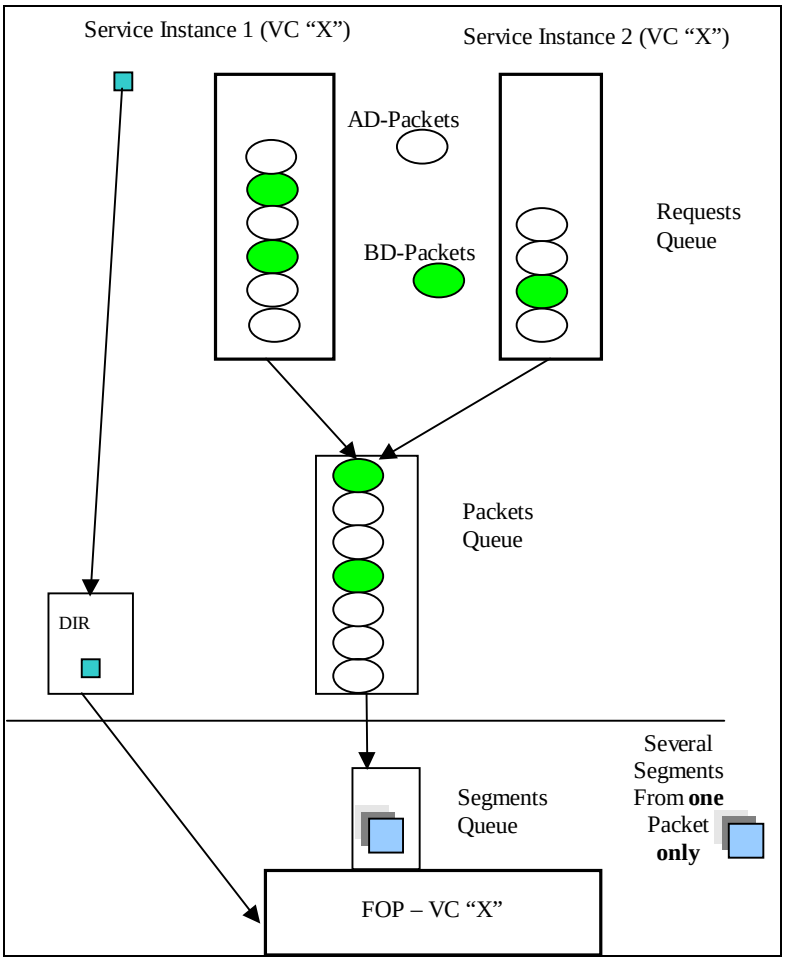


Figure B-3: Queues with MAP Multiplexing of FIFO Type

Any Packet in the Packets Queue is not yet in the state 'processing'[1]; i.e., it will be discarded following an FSP-STOP[2]. The segments still awaiting transmission are not discarded. The directive 'Reset VC' can abort any activity on the given VC and can clear all buffers.

When the Segments queue is empty, the Packets Queue is polled to fetch a Packet to be segmented. It should be noted that, whenever the number of unacknowledged AD and BC frames reaches the value of the FOP sliding window, transmission of BD will still be possible. This feature shall be exploited only for BD packets at the front of the Packets Queue. As

[1] Start of Processing **does not** mean Start of Transmission. In fact, (the first segment) of a Packet in the Segments Queue may remain there for a long time before being transmitted due to the MAP and/or TC Frame Multiplexing.

[2] After an FSP-STOP only Requests belonging to the issuing User shall be discarded.

soon as an AD packet is on the top of the Packets Queue no special search for further BD packets in the queue shall be performed.

B 3.2 Queues for Prioritized Multiplexing

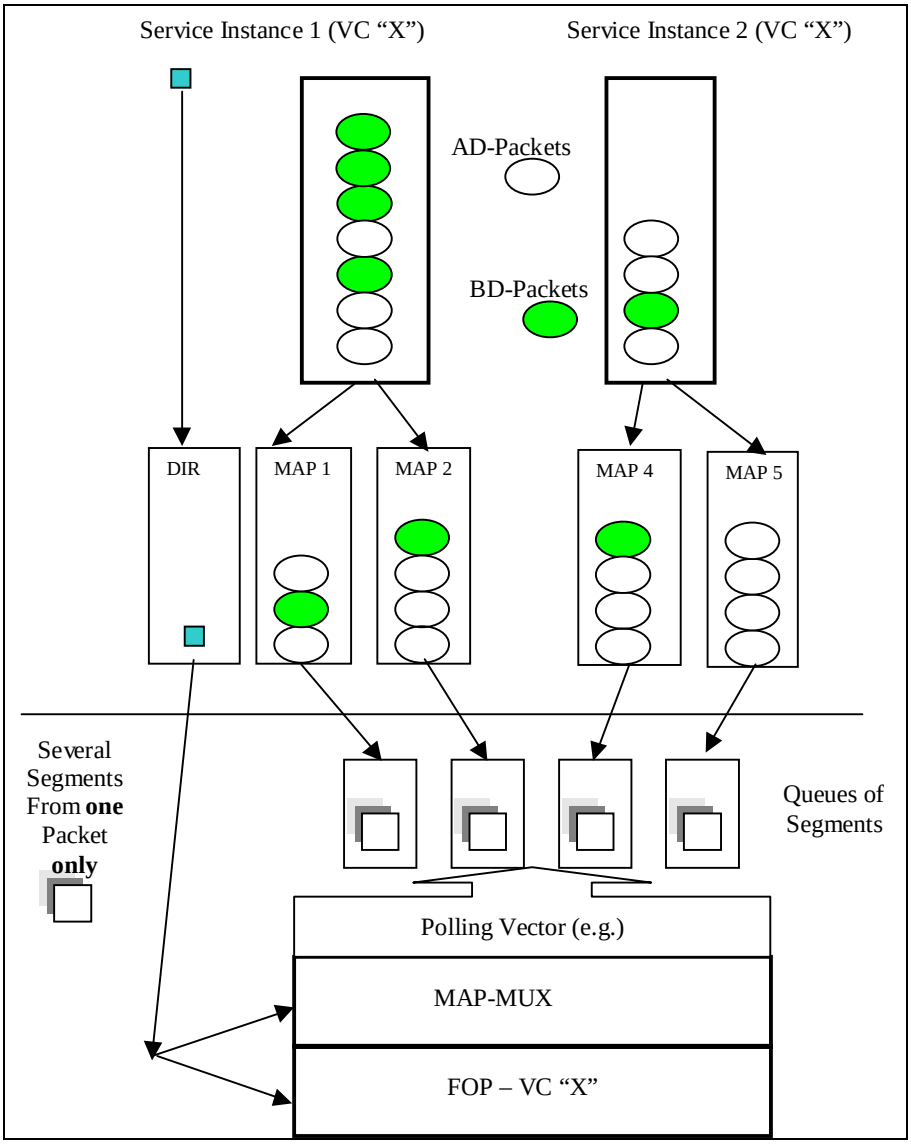


Figure B-4: Queues with prioritized MAP Multiplexing

Whenever a non-FIFO priority scheme is applied (either *Absolute Priority* or *Polling Vector*), more queues are logically required, as shown in figure B-4.

Each Service Instance still has its own Requests Queue. In this case, however, there will be a Packets Queue and a Segments Queue for each MAP. The Multiplexing Mechanism performs

polling on the several Segments Queues according to the applicable Multiplexing Scheme, e.g., Polling Vector, to fetch a segment to be passed to the FOP Machine for processing and eventual transmission.

Any Packet in the Packets Queue is not yet in the state ‘processing’ i.e., it will be discarded following an FSP-STOP. Further, the Packets in the Packets Queue are still subject to the check for the expiration of the ‘latest-production-time’. Should this time expire when the Packet is in the Queue, the Provider shall behave as described in this document (See FSP-ASYNC-NOTIFY). The Segments still awaiting transmission are not discarded. The Directive ‘Abort VC’ can abort any activity on the given VC and can clear all buffers.

ANNEX C

INDEX TO DEFINITIONS

(This annex **is not** part of the Recommendation)

This annex provides an index to the terminology defined in the references.

Telecommand, Summary of Concept and Service (reference [F2])

Term
Command Link Control Word
Command Link Transfer Unit
Command Operation Procedure
Physical Layer Operations Procedures

Cross Support Reference Model (Reference [1])

Term
Abstract-Object
Mission Data Operation System
Mission User Entities (MUE)
Service User/Provider
SLE Complex
SLE Utilization Management
Space Link Data Channel

ANNEX D

ACRONYMS

(This annex **is not** part of the Recommendation.)

This annex expands the acronyms used throughout this Recommendation.

ASDC	Abstract Service Definition Conventions, see reference [1]
ASN.1	Abstract Syntax Notation One
CCSDS	Consultative Committee for Space Data Systems
CLCW	Command Link Control Word
CLTU	Command Link Transmission Unit
CLTU-	CLTU service
CM-P	Complex Management Port
COP	Command Operation Procedure
DCE	Distributed Computing Environment
FG	Functional Group
ISO	International Organization for Standardization
MC-OCF	Master Channel Operational Control Field
MDOS	Mission Data Operation System
MUE	Mission User Entity
PDU	Protocol Data Unit
RF	Radio Frequency
SDU	Service Data Unit
SLE	Space Link Extension
SLE-FG	SLE Functional Group
TC	Telecommand
T-P	Transfer Provider Port
T-U	Transfer User Port
UTC	Universal Time Coordinated, see reference [F3].
VC-OCF	Virtual Channel Operational Control Field

ANNEX E

CONFORMANCE OPTIONS MATRIX

(This annex **is** part of the Recommendation.)

This annex provides the Conformance Matrix for implementations of Space Link Extension (SLE) Forward Space Packet (FSP) Service systems. An SLE FSP Service system will be considered to be 'conformant' if the mandatory elements identified in the matrix are implemented as described in this Recommendation.

Service Operations and their Parameters	**Optional / Mandatory**

a) FSP-BIND
b) FSP-UNBIND
	These operations are mandatory. All parameters are also mandatory with the exception of:
	`initiator-credentials`
	`responder-credentials`

c) FSP-START
d) FSP-STOP
e) FSP-TRANSFER-DATA
f) FSP-ASYNC-NOTIFY
g) FSP-SCHEDULE-STATUS-REPORT
h) FSP-STATUS-REPORT
i) FSP-SYNC-SET-PARAMETER
j) FSP-GET-PARAMETER
	These operations are mandatory. All parameters are also mandatory with the exception of:
	`invoker-credentials`
	`performer-credentials`
	`stop-production-time`

k) FSP-THROW-EVENT	This operation is optional.

l) FSP-INVOKE-DIRECTIVE	This operation is mandatory. All parameters are also mandatory with the exception of:
	`invoker-credentials`
	`performer-credentials`

m) FSP-PEER-ABORT	This operation is mandatory. All parameters are also mandatory with the exception of: `invoker-credentials`
n) FSP-PROTOCOL-ABORT	This operation and its parameter are mandatory.

Other Subsections	**Optional / Mandatory**
2.6.4: Buffering	Mandatory minimum size is 1024 (TBC) maximum sized FSPs
4.2: State transition table	Mandatory
Annex A: ASN.1 Protocol Specification	All data types are mandatory. Implementations may vary as described in the annex.

ANNEX F

INFORMATIVE REFERENCES

(This annex **is not** part of the Recommendation)

[F1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-7. Yellow Book. Issue 7. Washington, D.C.: CCSDS, November 1996.

[F2] *Telecommand Summary of Concept and Rationale*. Report Concerning Space Data System Standards, CCSDS 200.0-G-6. Green Book. Issue 6. Washington, D.C.: CCSDS, January 1987.

[F3] *Time Code Formats*. Recommendation for Space Data Systems Standards, CCSDS 301.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, April 1990.

[F4] *Space Link Extension Service Management Specification*. Draft Recommendation for Space Data System Standards, CCSDS 910.5-W-1.10. White Book. Issue 1.10. Silver Spring, Maryland, USA: CCSDS, 1997.

[F5] *Standard Terminology, Conventions, and Methodology (TCM) for Defining Data Services*. Report Concerning Space Data Systems Standards, CCSDS 910.2-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, November 1994.